A KNOWLEDGED-BASED GOAL-DRIVEN APPROACH FOR DECISION SUPPORT FOR OFFICE AUTOMATION

By

RODERICK D. WILSON

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1990

Copyright 1990

by

Roderick D. Wilson

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation and gratitude to the chairman of his supervisiory committee, Dr. Julius. T. Tou, for his technical advice and encouragement in carrying out this work. The author also acknowledges Dr. John Staudhammer for his valuable and sustaining words of encouragement throughout this endeavor.

To his remaining doctoral supervisiory committee members, Dr. Keith Doty, Dr. Fazil Najafi, and Dr. Zoran Pop-Stojanovic, the author expresses deep appreciation for their cooperation.

The author wishes to thank his family for their constant encouragement and understanding during all stages of study at the University of Florida, his friends and colleagues in the Center for Information Research (CIR) for providing a pleasant and cooperative working environment, and both the Florida Endowment Fund (FEF) for Higher Education and Bell Communications Research (Bellcore) for their support.

Finally, the author expresses special thanks to his wife, Anitra C. Wilson, for her many constructive, thoughtful, and insightful discussions, and for her support during all stages of matriculation at the University of Florida.

TABLE OF CONTENTS

page

ACKNOWLEDGEMENTS iii

ABSTRACT vi

CHAPTER

1 INTRODUCTION 1
1.1 Problem Motivation 1
1.2 Diversity of Office Tasks 6
1.3 Well-Structured vs Ill-Structured Problems . 7
1.4 Office Task Model 9
1.5 A Practical Role for Knowledge-based Systems in OA 18
1.6 From Mechanization to Automation 21
1.7 Organization of Dissertation 21

2 KNOWLEDGE BASE SYSTEM DESIGN METHODOLOGY 24
2.1 Knowledge-based Problem Spaces 24
2.2 Database vs. Knowledge base 25
2.3 Knowledge base Representation Formalisms . . 27
2.4 A Typical Knowledge base Architectural Overview 35
2.5 Rule-based Approach 40
2.6 Pattern-directed Approach 44
2.8 AUTOSEC High-level System Architecture . . . 46

3 AUTOSEC AND SYSTEM KNOWLEDGE FOR OFFICE AUTOMATION 49
3.1 Knowledge Categories for Office Automation . 49
3.1.1 Evolutionary Domain Knowledge 52
3.1.2 Evolutionary Organizational Knowledge 52
3.2 Knowledge base Structure and Representation . 53
3.2.1 Object Rulebase Organization 60
3.2.2 Symbolic Pattern Matching in the Object Rulebase 63
3.2.3 Task Partitioning Criterion 68

4 AUTOSEC KNOWLEDGE-BASED SYSTEM SEARCH AND CONTROL ISSUES 70
4.1 Distinguishing Between Search and Control . . 70
4.1.1 Depth First Search 73
4.1.2 Breadth-First Search 75
4.2 Office Task Problem Solving and AND/OR Trees 77
4.3 Bayesian Inference and Decision Making . . . 85
4.3.1 Relative Frequency 85
4.3.2 Expert Knowledge 89

5 AUTOSEC SYSTEM INTEGRATION AND IMPLEMENTATION . 93
5.1 Office Task Knowledge Processing 93
5.2 Office Task Object-Rule Construction 97
5.3 AUTOSEC Goal-driven Inference Strategy . . 103
5.4 Data Structures for Office Task Decision Support 110
5.5 AUTOSEC System Architecture 115
5.6 Uncertainty and Knowledge base Task Parameters 118
5.7 Experimental Office Task Domain Application 124

6 CONCLUSION 133
6.1 Summary 133
6.2 Areas for Future Work 135

APPENDIX 136

REFERENCES 155

BIOGRAPHICAL SKETCH 159

Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

A KNOWLEDGE-BASED GOAL-DRIVEN APPROACH FOR DECISION SUPPORT FOR OFFICE AUTOMATION

By

Roderick D. Wilson

December 1990

Chairman: Dr. Julius T. Tou
Co-Chairman: Dr. John Staudhammer
Major Department: Electrical Engineering

This study presents a novel design approach for knowledge-based system design for knowledge-based decision support for office automation which is a synthesis of knowledge engineering, pattern recognition, and artificial intelligence principles. The knowledge-based goal-driven approach we propose for office task decision support utilizes a goal frame tree schema for task organization, a hybrid rule-based pattern-directed formalism for task structural encoding, and a goal-driven inferential control strategy. Our system is a part of the knowledge-based system referred to as AUTOSEC which was proposed by Professor Tou.

The AUTOSEC knowledge base incorporates three basic structures to organize and control information: goal frames, task parameters, and an object-rulebase. Goal frames serve a

general organizational purpose, by defining a real or abstract problem in terms of goals and subgoals in the knowledge base. Task parameters are characteristic attributes of tasks in the form of information atoms that can be assigned one or more values. The object-rulebase consists of pattern/action assertional implications that describe the logical relationships existing between task parameter values. A goal realization process formulates symbolic pattern expressions whose atomic values map to task parameters contained a priori in a hierarchical database of task state information. Symbolic pattern expression creation is accomplished via the application of a novel goal-driven inference strategy which logically prunes an AND/OR tree constructed object-rulebase. Similarity analysis is performed via pattern matching of query symbolic patterns and a priori instantiated task parameters. An approximate degree of belief reasoning framework for handling uncertainty in knowledge-based task parameters is presented based on belief weighting factors.

The design approach we propose for knowledge-base system design for office task decision support can be applied to various tasks contingent on inclusive domain knowledge. Tasks can include those which incorporate individual knowledge aspects (IKAs) as well as organizational knowledge aspects (OKAs) such as meeting scheduling, personnel assignment functions, and financial expenditure recommendation.

Experimental results for some AUTOSEC task implementations are presented.

CHAPTER 1
INTRODUCTION

1.1 Problem Motivation

In the last two decades, our society has experienced a significant increase in the application and utilization of computers in performing and supporting different tasks. Two relatively new fields, knowledge engineering and artificial intelligence, have contributed to the building of computer systems capable of performing tasks like talking, planning, and analyzing visual scenes. Knowledge engineering is a field concerned with the study of the fundamental principles and the design of computer-based systems for the representation, organization, transfer utilization and extension of knowledge (Tou, 1985). Artificial intelligence is a field concerned with designing computer programs that exhibit intelligent behavior. When we describe people who perform similar task, we generally describe them in terms of what they know in order to perform the task. In other words, we describe someone's ability to behave with intelligence in terms of his or her knowledge. Similarly, we say that a computer program knows how to understand spoken English, or manipulate a robot. Thus, we ascribe knowledge to programs in the same manner as we ascribe it to people, based on observing

certain behavior. We ascribe knowledge to programs about objects in its domain, about events that have taken place, or about how to perform specific tasks. Our particular interest in the above discussion lies in the performance of specific tasks, particularly as they relate to office task decision support. In combining artificial intelligence methodologies with knowledge engineering and pattern recognition, the techniques employed can move well-understood human decision-making processes into the computer in the form of knowledge-based decision support systems for office automation.

Office task decision support is a key problem in the development of our society, since it involves a significant amount of workers and seems deemed to bring a considerable evolution in the organization and in the performance of office work. This field has attracted the attention of a very heterogenous crowd of researchers, and the resulting activity in this area has led to what we may call first generation office automation (OA) systems.

Recent years have seen the proliferation of many interesting tools for office workers. Graphical tools, text editors, database management systems, sophisticated spread sheets, and electronic mail are among the best known electronic tools. However, none of these tools deal with an approach that incorporates the "knowledge aspects" of the office environment for structured task decision support,

namely administrative knowledge, office knowledge and professional knowledge for problem solving.

To date, several methodologies and systems for office task automation have been proposed, most of which do not take into consideration the key information and knowledge germane in performing a given office task via the utilization of a knowledge-based paradigm for intelligent systems. Various approaches can be broadly classified into four basic model categories: 1) data-based models, 2) process-based models, 3) agent-based models, and 4) mixed models. Data-based models focus on grouping information and routing it by means of forms (Zloof, 1982; Jong, 1980; Trischritzis, 1979). Process-based models handle the representation of concurrent activities in order to provide execution and automation support for office work (Zisman, 1977; Ellis, 1979; Kunin, 1981; Ellis and Bernal, 1982). Agent-based models employ role and document objects to execute office tasks (Ellis and Bernal, 1982). Finally, mixed models employ a combination of both data-based and process-based models for automating office tasks (Bracchi and Pernici, 1983). The above approaches have focused on what we term as the "organizational aspects" of the office environment as opposed to the "individual aspects" for structured office task automation.

As the emerging field of artificial intelligence has gained practical acceptance, the utilization of artificial intelligence principles for office task automation is

beginning to receive attention in the literature and presents itself as a very challenging area of research (Barber, 1983; Croft and Leftkowitz, 1984; Maiochi and Pernici, 1987; Lochovsky, 1987). By viewing the nature of office work as a problem-solving activity, various techniques for providing decision support for specific office tasks have been presented. However, most of these approaches still employ deterministic models that are variations of the four previously presented. Most office tasks can seldom be modeled by a standard algorithmic solution to a well formulated problem. This is primarily due to the fact that office tasks exhibit a looseness of step-ordering. Thus, due the nature of the domain, most tasks cannot be modeled by the straightforward deployment of a sequential description method.

Contrary to what is still believed in most organizations, as well as institutions, independent of size, the key information needed in performing office-oriented tasks is not just a straightforward description of the steps that have to been taken, but moreover a knowledge of the goal and purpose of the tasks to be performed. In this context, what is generally considered to be "intelligence" can be organized and encoded into a collection of facts defining subgoals and a means of utilizing these facts via transformation rules to reach goals. A strong commonality exists among approaches for office task automation that employ AI techniques in that they lack a goal-oriented nature, in addition to a knowledge-based

control strategy, and for the most part exhibit a deterministic character (Croft and Leftkowitz, 1988; Trischritzis and Gibbs, 1987; Tueni et al., 1988). Thus, the fundamental problem: How does one organize, encode and control knowledge for computer-based decision support to aid in structured office task automation? To tackle this problem, we propose AUTOSEC, an intelligent decision support system approach for office automation which utilizes a novel knowledge-based goal-driven paradigm applied to rule-based knowledge system formalism. The AUTOSEC approach is based on the realization that in order to have more flexible systems for office task decision support, the representation of office tasks should include a description of the goal(s) of the activity involved, and artificial intelligence and pattern recognition problem-solving techniques to support achieving these goals.

Although some office tasks appear simple and highly structured, they are generally difficult to fully automate, due to the interactive nature of tasks performed in this domain. In short, even simple and structured tasks require sophisticated tools to model tasks at different levels of abstraction and to tailor control structures in order to handle the open-ended environment of this domain.

1.2 Diversity of Office Tasks

According to Mazer (1987), a primary description of office work considers the degree of structure of knowledge processing involved to perform the work, described as ranging from the routine to nonroutine. The diversity spectrum is also described as from structured to nonstructured, programmed to nonprogrammed, procedural to nonprocedural, algorithmic to problem solving, and most recently, low level to high level.

The location of task knowledge processing within this spectrum depends upon the degree to which the processing that must be performed to transform input to output can be specified independently of the intended goals of the processing activity. Routine procedural knowledge processing assumes that all the actions required to achieve goals can be captured and automated for all situations in which the processing will be done; the processing therefore embeds the goals implicitly into the program in the form of actions specified. Non-routine knowledge processing tasks acknowledges that one cannot predict and capture either the range of situations that will be encountered in the processing or the extent of the knowledge, activities, and resources that will be needed to carry out the intended tasks. The goals of the activity must therefore be represented explicitly to allow for adaptive, problem-oriented action to achieve goals.

Most knowledge processing belongs somewhere between the two extremes: for some situations, procedures can perform the

knowledge processing required; in other situations, goal structures must be used to handle unexpected contingencies. The latter occurrence is reflected in the office environment.

1.3 Well-Structured vs Ill-Structured Problems

Patrick and Fattu (1986) observed that well-structured, recurring problems occur in several domains. Well-structured means that there are a fixed number of features and a fixed number of categories. Such problems are named subsystems. To be useful, a subsystem should have recurring application and the knowledge should be recurring. In this sense, a subspecialist trained in the particular discipline or activity under study can transfer his/her knowledge expertise to a particular task.

Simon (1973) suggested that any problem solving process will appear ill-structured if the problem solver is a machine that has access to a very large long-term memory of potentially relevant information.

He further suggested that there is merit to the claim that much problem solving effort is directed to structuring problems; and only a fraction of it at solving problems once they are structured.

Further, there may be nothing other than the size of the knowledge base to distinguish ISPs (Ill-Structured Problems) from WSPs (Well-Structured Problems).

Simon suggested criteria for designing a WSP but stressed that any definition of a WSP may not be complete. With this in mind, we discuss the AUTOSEC approach. First, we define a state as a condition with respect to structure or circumstances.

A WSP in relation to the design strategy of AUTOSEC we employ has the following:

(1) Initial States: the knowledge base consisting of goal frames, task parameters (subgoals), relationships (object-rulebase), and belief measures (belief weighting factors)

(2) Goal State(s): goal parameter(s) for task completion

(3) Attainable State Changes: object-rules utilized to dynamically direct state transitions

(4) Knowledge (problem space): administrative, professional, and office knowledge as object-rules, task parameters, and goal parameters

(5) External World (know how to use it): the external world provides input through a particular task performed in the office environment and output through actions.

(6) Practical Amount of Computation: the use of microcomputers (as in AUTOSEC) or future microcomputers or new alternate methods of computing (Safayeri et al., 1987; Kratzer, 1987)

In the next section we present the theoretical foundations for a knowledge-based goal-driven model for office task decision support. Our model is based on the premise that

office tasks are goal oriented in nature, and as such, are inherently made up of subgoals that we define as task parameters for subsequent incorporation.

1.4 Office Task Model

In order to ameliorate the problem described in the previous section, the model we propose for knowledge-based decision support is structured around the explicit representation of the goals and subgoals prescribed by the task activities.

We view work that is performed in an office as consisting of various tasks. Each task can be decomposed into one or more subtasks. These subtasks may be further decomposed, resulting in a hierarchical task structure. This hierarchical view seems natural for office tasks and is an extension of those that can be found in other systems that support office tasks (Wier, 1984; Maes,1884; Tueni et al., 1988).

We view tasks performed by individuals within an office as goals. Each goal can be decomposed into one or more subgoals. These subgoals may be further decomposed resulting in a hierarchical goal-subgoal structure that describes the goal-subgoal relationship required to perform a given task. Mathematically, we know that the whole is equal to the sum of its parts, so by analogy we say that the goal of a task is defined by the disjunction or conjunction of subgoals. Figure 1.1 shows how a meeting scheduling task can be described as

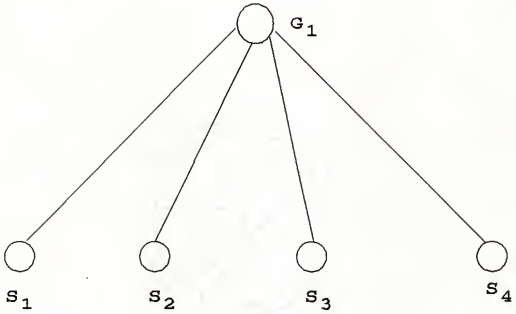
having a goal, which is the actual meeting schedule, and a set of subgoals that comprised the actual meeting schedule such as the time, date, location, participants, etc. Another practical example of office task goal decompostion would be personnel decision support for hiring decisions. The goal is to place a job applicant in an appropriate position, based on subgoals that describe an applicants qualifications such as level of education, previous work experience, grade point average, etc.

Let us formally denote a set of goal frames by C, and a set of task parameters by X,

$$X = \{x_i\},\ i = 1,\ldots,n.$$

A goal frame, C, is a general organizational structure that defines a real or abstract problem in terms of goals and subgoals in the knowledge base. We define task parameters as characteristic attributes of tasks in the form of information atoms contained in goal frames that can be assigned one or more values. For example, for a meeting scheduling task, the goal is the actual meeting schedule and the task parameters are the date, time, location, and participants, etc., for the meeting. Thus, task parameters are components contained in a set of object-rules in the object-rulebase OR, such that,

$$x_i \in C \text{ with } OR \subseteq C.$$



G_i - Goals
S_j - Subgoals $\qquad G_1 = (S_1, S_2, S_3, S_4)$

Figure 1.1 Goal-subgoal hierarchical task structure

Object-rules are pattern/action assertional implications that describe the relationships existing between task parameter values. We denote a set of task parameter values by S,

$$S = \{s_j\} \ , \ j = 1, \ \ldots, \ m$$

where $s_j \ \epsilon$ OR. We define a mapping function Q, which maps task parameter values to task parameters:

$$Q\colon S \rightarrow X \ (\text{ex. } x_1 = \text{time}, \ Q(s_1) = 9{:}30\text{am})$$

such that

$$Q(s_j) = x_i, \ x_i \ \epsilon \ C.$$

The function Q from S to X assigns an unique task parameter x_i of X to each task parameter value (subgoal) s_j of S. Figure 1.2 shows a typical mapping between task parameters and task parameter values defined by subgoals. The combinational union of task parameters, object-rules, and goal frames define the task problem space P^*:

$$P^* = X \ U \ OR \ U \ C$$

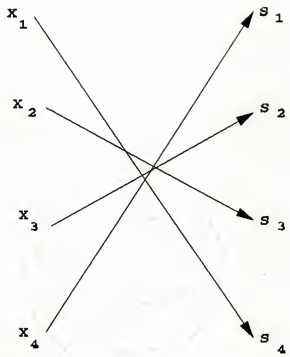
such that each task can be viewed as a distinct component oriented problem, thereby providing a generic structure for task formulation.

We impose the constraint that the set of goal frames C,

$$C = \{c_k\}, \ i = 1, \ldots, \ l$$

where l denotes the total number of goal frames, is to be comprised of a set of mutually exclusive task categories (goal frame constraint)

$$c_1 \cap c_2 \cap c_3 \ \ldots . \cap c_i = \emptyset, \quad \forall i.$$



X_i - Task parameters
S_i - Task parameter values
$Q: S \rightarrow X$

x_1 = month	s_1 = 00:30
x_2 = day	s_2 = 12:00
x_3 = hour	s_3 = 21
x_4 = minutes	s_4 = October

Figure 1.2 Bijective (one-to-one and onto) mapping of task parameters and values

The explicit representation of the goals prescribed by tasks can be described as a subgoal pattern formulation to goal realization process as follows:

$$\forall C,\ \exists S = \{X \mid Q(s_1) \wedge Q(s_2) \wedge Q(s_3) \ldots \wedge Q(s_j) \rightarrow G$$

For all goal frames, there exists a set of task parameter values that map to task parameters, such that the conjunction of task parameter values form a pattern to be matched in order to realize a goal. In the above equation the symbols utilized correspond to the following:

G - goal parameters
$X = \{x_i\}$ - set of task parameters
$S = \{s_j\}$, subgoals of task parameter values,
s_j - jth subgoal corresponding to goal G
x_i - ith task parameter corresponding to the jth subgoal
Q - bijective mapping function
$\rightarrow$ - realization
$\wedge$ - conjunction

KNOWLEDGE-BASED GOAL-DRIVEN MODEL VALIDATION EXAMPLE

Goal frame: c_1
A priori instantiated task parameters: x_3, x_4
Object-rulebase:
OR_1: $x_4 \wedge Q(s_2) \wedge Q(s_5) \rightarrow Q(s_6)$
OR_2: $Q(s_7) \wedge Q(s_2) \rightarrow Q(s_1)$
OR_3: $x_3 \wedge Q(s_6) \rightarrow Q(s_1)$
OR_4: $x_4 \rightarrow Q(s_8)$
OR_5: $Q(s_2) \rightarrow Q(s_5)$
OR_6: $Q(s_8) \wedge Q(s_1) \rightarrow Q(s_9)$
OR_7: $x_3 \rightarrow Q(s_2)$
OR_8: $Q(s_8) \wedge x_3 \rightarrow Q(s_1)$
OR_9: $Q(s_8) \wedge x_4 \rightarrow Q(s_2)$

where

$\wedge$ - conjunction
$\rightarrow$ - implication

Goal parameter: $G_1 = Q(s_9) = x_9$
MEANING: Does goal parameter x_9 follow from the instantiated task parameters using the given object-rules contained in goal frame c_1.

Goal realization equation:

$$
\begin{aligned}
G_1 &= \{x_9 \mid Q(s_8) \wedge Q(s_1)\} \\
&= \{x_9 \mid x_4 \wedge (x_3 \wedge Q(s_6))\} \\
&= \{x_9 \mid x_4 \wedge x_3 \wedge (x_4 \wedge Q(s_2) \wedge Q(s_5))\} \\
&= \{x_9 \mid x_4 \wedge x_3 \wedge x_4 \wedge x_3 \wedge (Q(s_5))\} \\
&= \{x_9 \mid x_4 \wedge x_3 \wedge x_4 \wedge x_3 \wedge x_3\}
\end{aligned}
$$

Given the previous goal frame, task parameters, object-rulebase (i.e. knowledge base) and goal parameter, the task is to find whether or not goal parameter x_9 can be realized using the given object-rules. The goal parameter x_9 occurs only in the consequence of object-rule OR_6, thus we have two new subgoals $Q(s_8)$ and $Q(s_1)$. Object-rule OR_4 has $Q(s_8)$ as a consequence, thus we have two more subgoals $Q(s_1)$ and x_4. Since x_4 is in the given task parameter set, we now have subgoal $Q(s_1)$. There are three object-rules OR_2, OR_3, and OR_8 that have $Q(s_1)$ as a consequence, so the AND/OR tree to be searched has three branches as shown in Figure 1.3. Branch 1 from object-rule OR_2 has two subgoals $Q(s_7)$ and $Q(s_2)$. Since $Q(s_7)$ is not a consequence of any object-rule, we are unable to proceed down this path. Branch 2 from object-rule OR_3 gives two new subgoals x_3 and $Q(s_6)$. Task parameter X_3 is given, thus we have a new subgoal $Q(s_6.)$ From object-rule OR_1 along branch 2 we have three new subgoals x_4, $Q(x_2)$, and $Q(s_5)$. The task parameter X_4 is given, thus we have two new subgoals $Q(s_2)$ and $Q(s_5)$. From object-rule OR_7 along branch 2 we have two new subgoals x_3 and $Q(s_5)$. Task parameter X_3 is given, thus the new subgoal is $Q(s_5)$. From object-rule OR_5 and object-rule OR_7 we have two subgoals, $Q(s_2)$ and x_3 respectively, and since x_3

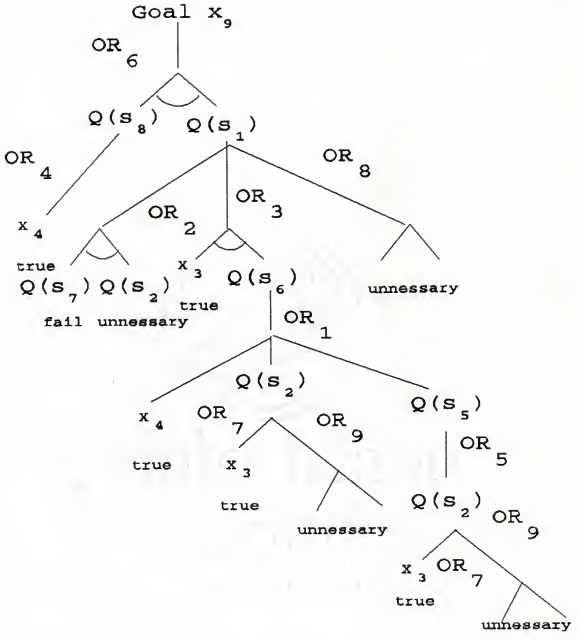


Figure 1.3 Hierarchical AND/OR tree goal-subgoal task decomposition for goal realization.

is a given task parameter, we have logically pruned a path that realizes goal parameter x_9 from the set of object-rules.

The function of this model in AUTOSEC is to logically prune an AND/OR tree constructed knowledge-base. For all goals defined in a specific goal frame there exists a unique and finite set of subgoals which are logically pruned in an AND/OR tree backward-chaining object-rule activation process which binds task parameters to specific task parameter values for use in what we term "goal realization." The goal realization equation is based on a hypothesis promotion process which consists of the firing of those object-rules in the knowledge-base that contain subgoals that instantiate (i.e. assign a task parameter a value) task parameters contained in the object rulebase. Task parameters are subsequently utilized in decision-making using rule-based pattern matching. The rule-based pattern matching technique uses the task procedural logic contained a priori in the task knowledge base. Our model through the incorporation of pattern recognition and artificial intelligence via problem reduction techniques represents a new approach for knowledge-based system decision support for office tasks. The goal-driven control strategy we employ is analogous to a decision theoretic approach which can viewed as a rule (Wilson, 1986):

TASK DESCRIPTION → GOAL FRAME

which assigns a task description ("event") to a goal frame.

In the next section we discuss the practical role of a knowledge-based decision support system for the office environment via the description of several potential task domain applications.

1.5 A Practical Role for Knowledge-based Systems in OA

Most work related to the utilization of knowledge-based approaches for various aspects of office automation has primarily been concerned with the improvement of organizational effectives and efficiency. We view the role of knowledge-based systems in office automation to be one of providing a means whereby problem-solving and decision-making processes currently handled by individuals can be assisted by intelligent machines. This point of view has at its foundation the increasing benefits of computer memory capacity expansion, software sophistication and the lowering of hardware costs.

Consider as an example an office task as ubiquitous as personnel time management. In scheduling meetings, an individual must interact with others in order to determine a generally acceptable (though not "optimal") time. Depending on the structure of the organization, the interaction may take place as a form of negotiation among of individuals until they converge on an agreeable time. Alternatively, all subordinates may be required to submit available times to

coordinators, who then in turn unilaterally decide upon a time. Note that individuals in the latter case have less immediate control over the answer sythesis (i.e. the time selection), and this may effect the way they construct the list of available times. For example, though a coordinator indicates the meetings will take place only one hour, meetings with this coordinator are known to take two hours, so the subordinates submit only two-hour-block times as a defensive measure.

Consider a system for supporting personnel assignment decisions in an organization. You are the personnel director for a large engineering firm, and an applicant, resume' in hand, comes to you for an interview. The applicant's qualifications are before you, as are the requirements of the job and the general standards of the company. Your problem is which position, if any, is this applicant qualified for?

This may not seem like a very complicated problem at first glance, but there are many factors that go into such a decision. Suppose the applicant has little experience in engineering, but made an important discovery in his or her field. Or suppose the applicant did not do well in school but has several years of solid work experience. Each applicant is different, and although certain criteria must be met to land a job, there are several different combinations of factors in any applicant's history that may make that person qualified for a particular position.

Consider a third problem pertaining to financial expenditure planning based on institutional or organizational office budgetary constraints, wherein the specifics of the problem centers around determining whether or not leasing or purchasing particular services or equipment is the most appropriate decision.

In each of the above examples, a degree of interaction is required in order to accomplish the desired tasks. Each task can be described as consisting of goals, limited domain knowledge, knowledge of the connections among the goals (relations), and of the authority of the relationships among them.

Thus it can be observed, that the final result to which all our thought process are directed are goals. Even when engaged in the most simple physical task or the most complex mental activity, the mind is sharply focused on a goal. Without goals, we have no reason to think.

When designing knowledge-based systems the goal of the system must always be kept in mind. We don't do things because we think, we think because we have things to do. We all know that the human mind possesses a vast store of knowledge relating to a countless array of objects and ideas. Survival depends on our ability to apply this knowlege to any situation that arises and to continuously learn from new experiences so that we will be able to respond to similar situations in the future. Our interest in the previous

discussion lies in the ability to incorporate knowledge into systems for applications to the office enviroment.

1.6 From Mechanization to Automation

The support of office work with computers has evolved significantly in its relatively short history. The recent shift from mechanization to automation is reflected in three ways: 1) the computer plays a more active role in initiating and controlling work; 2) the integration of more functions is provided and supported by the computer; and 3) the support is no longer centered on tools for individuals, but rather on functions performed, often by many individuals, with differing responsibilites and levels of expertise.

Currently, the term "office automation" currently refers to the use of diverse information and technology in support of business processes. Since we place emphasis on the support role of computers in organizations, we introduce the the phrase knowledge-based office support systems (KBOSS).

1.7 Organization of Dissertation

In Chapter 2, we begin by presenting an overview of knowledge base system methodology. This chapter provides an architectural overview of system components, knowledge representation formalisms, and design approaches. We describe the difference between a database and a knowledge base,

followed by the four major components of a knowledge-based system, namely the knowledge base, knowledge acquisition mechanism, recognition/inference mechanism, and user interface system. We investigate knowledge representation formalisms and their role in intelligent system knowledge for office automation. We discuss the merits and disadvantages of four major knowledge representation schemes, namely, mathematical logic, semantic networks, production rules and frames. We continue with a discussion of two major design approaches, specifically the rule-based approach and the pattern-directed approach. We conclude with a high-level description of the AUTOSEC system architecture.

Chapter 3 presents issues related to system knowledge for office automation. We begin with our definition of knowledge categories for office automation. In addition to the proposed method of knowledge base structure and knowledge representation we employ for object-rule construction, we describe task partitioning criterion, and symbolic pattern matching in the object rulebase.

Chapter 4 is concerned with knowledge-based search and control issues. We begin by distinguishing between search and control. We subsequently develop a strategy that combines office task problem solving with AND/OR trees as a method of object-rulebase construction. We conclude this chapter by discussing the major limitations of applying a Bayesian approach for inferential analysis for office automation.

Chapter 5 is centered around the AUTOSEC knowledge base system integration and implementation. We propose a knowledge-based formalism for office automation, which incorporates a goal-driven inference control strategy and implements our office task model. In addition, we describe the major components in an architectural sense of our approach in a system architecture for a knowledge-based system for intelligent decision support for office automation, along with data structures for office task decision support. We conclude this chapter by discussing issues related to handling uncertainty in knowledge-based task parameters using belief weighting factors and give an example application.

Finally, in Chapter 6 we summarize our research contributions and give suggestions for future research.

CHAPTER 2
KNOWLEDGE BASE SYSTEM DESIGN METHODOLOGY

2.1 Knowledge-based Problem Spaces

The design of a knowledge-based system requires not only knowledge about the domain, but also problem solving capacities. Knowledge-based system design is a goal-driven task which can be analyzed according to the problem space principle (Card, Moran, Newell, 1983). However, according to Rappaport (1986), the problem space on which this problem solving task is performed can be said to be made of three subspaces: 1) the methodological space, 2) the application space, and 3) the processing space. The methodological space consists of the knowledge from various methodologies to achieve a given goal. The application space is comprised of different elements of knowledge representing the various states of the problem and operators leading from one state to another. The processing space is the space generated by the combination of the two others, during execution. As in all rational tasks, there is a permanent relative use of search or knowledge. While the knowledge base may have a deep knowledge of the domain, formalization of this knowledge is difficult. The identification of different problem spaces facilitates and

allows the analysis of the task to be performed and aids in the design of the knowledge-based system components.

Before giving an overview of the typical components of a knowledge-based system and knowledge representation formalisms, we will first begin by providing a conceptual foundation for knowledge base system design methodology.

In the next two sections, we discuss the distinguishing characteristics of a database versus a knowledge base, and give an overview of some generally used application-oriented knowledge representation formalisms.

2.2 Database vs. Knowledge base

A database can best be described as a collection of data representing facts. The amount of data is typically large, and the facts change over time. The corresponding updates to the data can be made routinely, often by clerical personnel. The correctness of the facts can be determined objectively by comparing the data values with real world observations.

Databases are typically maintained for operational purposes, frequently for resource control. Such databases contain inventories of material and personnel. The predominant way of interacting with databases is through programs and report generators. In databases, these objects are represented by data in the database.

In contrast, a knowledge base, contains information at a higher level of abstraction. Knowledge-based systems are of

interest in problem areas difficult for algorithmic methods. These systems describe and operate on classes of objects rather than on individual objects.

The contained knowledge in a knowledge base is typically generated by experts having knowledge about some domain related to the database of the database system. This knowledge is typically descriptive, relates to general aspects of the data, and is significantly smaller than the data. Knowledge should not vary rapidly over time, like data, since changes might require mediation by an expert or some other form of correctness verification. The knowledge in a knowledge-based system is used typically for data analysis and planning tasks which are usually performed by decision-makers within an enterprise.

In order to better illustrate the differences between knowledge and data in a practical sense, suppose we are given four assertions: (1) Mike is 43 years old. ¦DATA¦; (2) Middle age is 40 to 60 years ¦KNOWLEDGE¦.; (3) People of middle age are careful ¦KNOWLEDGE¦.; (4) Mike has never had a traffic accident ¦DATA¦. If we have the query: (Q) Is Mike careful?. We can look at facts (in this particular instance assertion 4) by wording the actual query appropriately or we can deduce an identical result from the knowledge-base. In this case, "yes" can be deduced from assertions 1, 2, and 3.

Knowledge is used to control the database. The information that "age" is an attribute of "people" and that

"middle age" is an age is intensional knowledge for the database. Intensional knowledge is defined as knowledge beyond the factual content of the database.

2.3 Knowledge base Representation Formalisms

As stated in Chapter 1, artificial intelligence research is concerned with creating programs that exhibit intelligent behavior, and knowledge engineering is concerned with putting knowledge and information to work for people. In order to design systems that exhibit intelligent behavior and also put knowledge and information to work for people, schemes have been developed for incorporating knowledge about the world into computer programs. These knowledge representations techniques involve routines for manipulating specialized data structures to make intelligent inferences. Most of the research in artificial intelligence assumes that what needs to be represented is known a priori. Thus, the task becomes one of figuring out how to encode the knowledge and information into the systems' data structures and procedures.

Techniques and theories about knowledge representation have undergone rapid change and development in the last few years. The understanding of different representation schemes that researchers have devised is one of the most active area of artificial intelligence research at present (Barr and Feigenbaum, 1981; Tanimoto, 1987). The following sections

will discuss the strengths and weaknesses of several extensively used knowledge representation formalisms for knowledge transfer and utlilization by computer.

2.3.1 Mathematical Logic

The classical approach to representing knowledge about the world contained in sentences like "ALL EMPLOYEES HAVE RESPONSIBILITIES." is a formal logic developed by philosophers and mathematicians as a calculus for the process of making inferences from facts. Translation of the above sentence into a mathematical formula yields

∀ x. EMPLOYEES(x) ----> (have(RESPONSIBILITIES(X)));

which reads, for any object x in the world, if x is an employee, then x has responsibilities. The advantage of formal representation is that there is a set of rules called the rules of inference in logic, by which facts that are known to be true can be used to derive other facts known to be true. For example, suppose we add another fact to our knowledge-base:

∀ x. MANAGER(x) ----> EMPLOYEE(X);

which reads, for any object x in the world, if x is a manager, then x is an employee. It follows that from these two facts we can conclude, using the rules of inference, that the following must be true:

∀ x. MANAGERS(x) ----> (have(RESPONSIBILITIES(X)));

that is, all managers have responsibilities. Although logic-based representation formalisms have been used with some success in systems with relatively small databases, when the number of facts in the database increases, there is a combinatorial explosion in the number of ways to combine facts to make inferences when search is involved. Many problem domains (such as theorem proving) have an infinite search space, and the search space in others, though finite, is unimaginably large. Estimates of search-space size may be based on the total number of nodes (however defined) or on other measures. In chess, for example, the number of different complete plays of the average-length game has been estimated at 10^{120}, although the number of "good" games is much smaller. Even for checkers, the size of the search space has been estimated at 10^{40}.

The critical problem of search in logic is the amount of time and space necessary to find a solution. As the chess and checkers estimates suggest, exhaustive search in logic is rarely feasible for nontrivial problems. Examining all sequences of n moves, for example, would require operating in a search space in which the number of nodes grows exponentially with n. As mentioned previously such a phenomenon is called a combinatorial explosion.

Even though unrestricted resolution is robust, in that an answer will always be returned if one exists, logic is too indirect for nontrivial problems (Barr and Fiegenbaum, 1981).

Logic can be referred to as declarative knowledge. The major components consist of propositions, wich are properly formed statements with true or false values and predicates which are statements about individuals. There are two important quantifiers, FOR ALL and THERE EXISTS. First order logic refers to logical statements with quantification over individuals only. Second order logic is an extension of first order logic with the introduction of statements involving variables. For example, the statements such as "x > 3", "x = y + 3", and "x + y = z" are often found in mathematical assertions and in computer programs. These statements are neither true or false when the values of the variables are not specified. The statement "x is greater than three" has two parts. The first part, the variable x, is the subject of the statement. The second part, the predicate "is greater than three," refers to a property that the subject of the statement can have. In second order logic, we denote the statement "x is greater than three" by P(x), where P denotes the predicate "is greater than three" and x is the variable. The statement P(x) is also said to be the value of the propostional function P at x. Once a value has been assigned to the variable x, the statement P(x) has a truth value.

The reasoning process used in this representation formalism is based on propositional calculus, where propositions are combined by five sentential connectives: AND, OR, NOT, IMPLY AND EQUIVALENT. For example, "Employees can

receive reduced company insurance premiums, if service date is greater than five years AND health record is excellent." The following are some equivalent conversions used in logic:

$$X \wedge Y = Y \wedge X \quad \text{commutative law of conjuctions}$$

$$X \vee Y = Y \vee X \quad \text{commutative law of disjunctions}$$

$$\neg\neg X = X \quad \text{double negation}$$

$$\neg(X \wedge Y) = \neg X \vee \neg Y \quad \text{Demorgan's theorem}$$

$$\neg(X \vee Y) = \neg X \wedge \neg Y \quad \text{Demorgan's theorem}$$

$$X \wedge (Y \vee Z) = X \wedge Y \vee X \wedge Z \quad \text{Distributive law}$$

The major advantages of logic-based formalisms are that they are precise, flexible, and exhibit good modularity in terms of implementation. The major disadvantages are that the representation is separated from processing, heuristics are difficult to implement, and only exact type of reasoning is supported.

2.3.2 Semantic Networks

The semantic network was invented as an explicitly psychological model of human associative memory. A network consists of nodes representing objects, concepts, events, and links between nodes that represent their inter-relations (Winston, 1979). Both the nodes and links between them can have labels. Figure 2.1 illustrates a semantic network. One key feature of the semantic network is that important associations can be made explicitly and succintly, such that relevant facts can be inferred from the nodes to which they

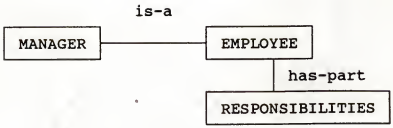


Figure 2.1 An example semantic network

are directly linked, without an exhaustive search through a large database. The interpretation or semantics of network structures depend solely on the program that manipulates them since there are no formal conventions about their meaning. Therefore, inferences drawn by manipulation of the network are not assuredly valid in the sense that they are assured to be valid as in a logic-based representation scheme. The major advantages of semantic networks are that the node and link schemes capture the essential associations between symbols and that it is easy to represent property inheritance. The major disadvantages are lack of formal semantics and too simplistic to represent non-trivial problems.

2.3.3 Production Rules

A production system is a modular knowledge representation scheme that is finding increasing popularity in large programs designed to exhibit intelligent behavior. The basic idea is that the knowledge base consists of rules, called productions, in the form of condition-action pairs:

IF (condition --> true), THEN (action --> appropriate)

The utility of this formalism comes from the fact that conditions in which each rule is applicable are explicit and, in theory at least, the interactions between rules are minimized, since one does not call another. During the execution of the production system, a production whose action

part is satisfied can fire, that is, have its action part subsequently executed by the interpreter.

A production system consists of three fundamental parts: 1) a rule-base composed of a set of production rules; 2) a special buffer-like data structure called the data memory; and 3) an interpreter. The data memory is the focus of attention of the production rules. The left-hand side of each production in the rule-base represents a condition that must be present in the data memory buffer before the production can fire. The actions of the production rules can change the data memory buffer, so that other rules will have their condition part satisfied. The data memory may be a simple list, a very large array, or more typically, a medium-sized buffer with some internal structure of its own.

The interpreter is a program whose job is to decide what to do next. In a production system, the interpreter has the special task of deciding which production rules to fire next. Production systems operate in cycles, with each cycle consisting of three phases: 1) matching; 2) conflict resolution; and 3) action. In the matching phase of each cycle, the productions are examined in a manner specified by by the interpreter to see which are appropriate and can possibly fire. The second phase of each cycle, conflict resolution, selects the most appropriate single production rule if more than one is appropriate. The third and last phase of each cycle, action, fires the production. Because

productions facilitate human understanding and the modification of systems with large amounts of data, they have been employed in several large knowledge-based systems like MYCIN (Shortliffe et al., 1977; Buchanan, 1978). The advantages of production systems lie in their modularity and naturalness. The disadvantages are inefficiency and opacity.

2.3.4 Frames

A frame is a knowledge representation formalism for representing a "stereotyped or well structured situation". Its power lies in the organization aspects of its utilization, in that both declarative and procedural knowledge can be incorporated. Frames have expectation-driven reasoning, property inheritance, extensive embedded procedural knowledge, based on if-added, if-removed, and if-needed procedures. The advantages of frames lie in their efficiency, flexibility and adaptivity. The disadvantages lie in the lack of a unified theory and general structure.

2.4 A Typical Knowledge base Architectural Overview

A knowledge-based system generally consists of four fundamental components: knowledge-base, knowledge acquisition mechanism, recognition/inference mechanism, and a user interface subsystem (Tou 1984, Tou 1985). The organization of a knowledge-based system is shown in Figure 2.2. This system

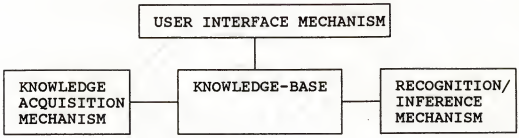


Figure 2.2 Architectural overview of a knowledge-based system

organization has successfully been employed in several systems developed in the Center for Information Research (CIR).

2.4.1 Knowledge base Description

The knowledge base is composed of a knowledge sketch and knowledge details. The concept of a knowledge sketch is used to represent the relational structure and the hierarchies of the problem orientation of the knowledge that is stored.

An associative tree hierarachical structure that employs semantic links is one way of representing knowledge in a natural format. In this structure, the most general items are placed at the top of the tree and the most specific items are placed at the bottom of the tree. The structure of a knowledge base provides useful information to facilitate knowledge classification, and new knowledge acquisition via insertion at the most appropriate location in the tree, identification of possible problems, and efficient seeking of appropriate knowledge in response to a query for intelligent decision-making (Tou, 1985). The knowledge tree hierarchy is linked to knowledge details via semantic pointers. The basic primary contents in a knowledge-base are definitions, characteristics, methodologies, causal relations, rules of thumb, transfer functions, performance goals, and reasoning procedures.

2.4.2 Knowledge Acquistion Mechanism

The knowledge acquisition mechanism automatically generates the knowledge base after a body of knowledge on a subject matter has been acquired and entered into a computer. Knowledge base generation is performed in an interactive conversational mode for office automation. The interactive mode is used in conjunction with the user interface system described in a later section. The knowledge base is divided into two components: knowledge sketch and knowledge details. The task of generating the relational hierarchy for the knowledge sketch may be accomplished by taking what is referred to as the bottom-up approach or the top-down approach. In the bottom-up approach, it is assumed that a node in the hierarchy will contain all the characteristics which would allow this aggreagation. The top-down approach is based upon the decomposition of upper-level nodes in the hierarchy. The selection of either approach depends on the system design and knowledge seeking strategies.

2.4.3 Recognition/Inference Mechanism

The recognition/inference mechanism component of a knowledge-based system interprets the user's query input, performs pattern matching, issues specific answers, and generates recommendations or inferences. The decision-making process is accomplished by discriminant analysis, Bayesian

statistics, clustering techniques, feature extraction, syntactic rules, and production rules. The major tasks performed by the recognition/inference mechanism are diagnosis, data validation, consistency analysis, violation detection, decision simulation and pattern recognition. These tasks are accomplished by a combination of pattern matching, content searching, rule intrepretation, mathematical operations, inference heuristics, and programmed algorithms (Tou, 1985). In the next section we give an overview of the characteristics and operational responsibilities of the user interface component of a knowledge-base system.

2.4.4 User Interface System

The major function of the user interface system in a knowledge-base system for office task decision support is to accept user dialogue in an interactive conversational manner. Most previous work in the Center for Information Research, regarding the design of the user interface utilized telebrowsing concepts. These concepts were developed and successfully used in a library retrieval system (TBS) (Tou,1976) and medical knowledge system (MEDIKS) (Tou, 1978) However, for office task decision support, straightforward telebrowsing techniques are not adequate for solving problems that involve a significant amount of symbolic manipulation for intelligent decision-making. Our design philosophy is to make the user interface of the knowledge base system for office

task decision support a very important component of the system due to the interactive nature of office tasks.

The next section describes one of the two major approaches for designing a knowledge base system. The two approaches include a rule-based approach and pattern-directed approach (Wilson, 1986; Tou 1985).

2.5 Rule-based Approach

In designing a knowledge-base system using the rule-based approach, the system will consist of two primary components: a rule-base and a rule interpreter (Buchanan and Shortliffe, 1984). The rulebase consists of rules which specify general knowledge about the problem domain. A rule in the rule-base is a conditional sentence relating several factual statements in a logical fashion. The condition part of a rule may be considered as any pattern that can be matched against the database. Once a match is made, the action part can be executed.

The rule interpreter may be considered as a kind of recognition/inference mechanism, which is designed to perform the following tasks (Tou, 1984): 1) decide the order in which rules are applied; 2) evaluate and apply a single rule; 3) determine how the condition of a selected rule should be matched to the database; and 4) monitoring the problem-solving process.

The problem solving process that is monitored by the rule interpreter for a knowledge-based system for office task decision support consists of three phases: 1) task knowledge acquisition; 2) task problem identification; and 3) recommendation or conclusion generation. See Figure 2.3. In the knowledge acquisition phase, knowledge is obtained which relates to a particular problem or task to be performed. In the problem identification phase, this knowledge is subsequently employed to identify the problem. In the recommendation or conclusion generation phase, the methodologies and procedures for solving the problem are determined.

In designing a knowledge-base system that ulitilizes pattern recognition and artificial intelligence techniques, the rules have the following form:

PATTERN ----> ACTION

The strategy used by the rule interpreter is often called the control strategy. The commonly used control stragies are data-driven, goal-driven, and mixed strategies. In the data-driven control strategy, rules are applied whenever their left-hand side conditions are satisfied. Goal-driven control or backward chaining is the strategy which does not seek information nor apply rules that are unrelated to the overall goal. A weighting factor reflecting the importance or probability can be assigned to a rule if the left-hand side condition can be satisfied in more than one way.

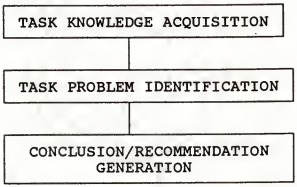


Figure 2.3 AUTOSEC problem solving process

The knowledge-base is made up of a global database, which contains facts or assertions about the particular problem being solved, and a rule-base. The global database is analogous to the knowledge details, and the rule-base is analogous to the knowledge sketch. In the global database, factual knowledge, described in the next chapter, may be represented as in arrays, strings of symbols, list structures, etc.

In addition to the knowledge-base and rule interpreter, a knowledge base system designed using the rule-based approach also contains an auxiliary memory for storing intermediate results of rules when they are actuated or fired. Such an auxiliary memory is often referred to as the sketch pad or blackboard (Waterman,1986).

Rule-based systems have evolved from a more general class of computational models known as production systems. Instead of viewing computation as prespecified sequences of operations, productions treat computation as the process of applying transformation rules in a sequence determined by the data thus performing recognition/action cycles. In each cycle the right-hand part of the rule is executed, after the rule interpreter first decides which rules to evaluate and apply. Several systems have been designed based on this approach, one of the most successful has been MYCIN (Shortliffe et al., 1977).

2.6 Pattern-directed Approach

A knowledge-base system may be designed on the basis of the principles of knowledge-based pattern recognition (Tou, 1982; Tou, 1984; Tou, 1985; Dai and Tou, 1988). In the past, pattern recognition studies focused on a relatively small number of pattern classes that were characterized by a small set of features. However, in recent years, attention has turned to a second type of pattern recognition problem which deals with very large numbers of pattern classes characterized by extremely large sets of attributes. Approaches to solving this problem utilize a knowledge base and are referred to as knowledge-based pattern recognition. The MEDIKS (Chang and Tou, 1984) system was designed based on this approach.

The knowledge base in a pattern-directed system consists of a knowledge sketch and knowledge details. As opposed to the knowledge sketch in the rule-based approach, the knowledge-sketch in the pattern-directed approach is characterized by three basic primitives: 1) entities; 2) attributes; and 3) relationships.

The knowledge sketch is represented by a hierarchy of entity-attribute-value relations. This structured knowledge is semantically categorized into an associative tree structure. Each node of this relational hierarchy represents an entity (goal) associated with a set of attributes (task parameters) and values which form the information pattern describing a knowledge domain.

Both the pattern-directed and rule base approach make use of the idea of pattern matching. However, one of the main differences between the two approaches lies in the method of knowledge representation employed. The main distinction being that the knowledge structure in a discipline is fully utilized in the design of the knowledge base for pattern-directed knowledge-based systems (Suh, 1981; Tou, 1978; Tou, 1985).

In our knowledge-based system design for office task decision support, knowledge is encoded into knowledge task parameters and relations defined by object-rules. Object-rules act to encode the relationships between attributes and entities. In this dissertation, parameters associated with tasks (task parameters) and their values (task parameter values) are defined as attributes. Attributes refer to the characteristics of an entity. In the context of goals and subgoals, the parameters associated with a goal (goal parameters) are considered as entities and task parameters or task parameter values are considered subgoals. Attributes, when associated with a particular entity occurrence, may be valued or weighed. The characteristic attributes for each entity occurrence may be viewed as a pattern expression. This pattern expression may be used as a basis for machine intelligent decision making operations. The inter-relationships between knowledge entities are called knowledge relations or simply relations. In any significant knowledge base, the number of relations is likely to be practically

infinite. Relations in a knowledge base can be grouped into two classes: 1) those relations which can be processed by machine; and 2) those which only can be recognized by mankind. For example, a rule base or associative tree representation for a hierarchical classification may be recognized and process by machine. On the the other hand, a textual definition, such as a particular office task function may be retrieved and displayed to the user but is of little use for automatic decision making (Wilson, 1986; Tou, 1984).

2.8 AUTOSEC High-level System Architecture

Based on the model previously described in chapter 1 section 1.4 we have proposed a high-level block schematic of a knowledge-based architecture for decision support for office task decision support which is shown in Figure 2.4. This architecture consists of five major components: 1) user interface-mechanism; 2) goal frame schema; 3) object rulebase ; 4) hierarchical database of task parameters; and 5) the control scheme or interpreter. The interface-mechanism is the human to machine interface mechanism. It can be a computer keyboard (local acess), telephone (remote access), or any other standard human to machine interface mechanism.

Barber (1983) emphasized the interactive nature of office-problem-solving and mentioned some differences between this environment and the typical environments handled by

intelligent problem-solvers. The main features of the office are that there can be many subtasks cooperating to achieve a goal and the knowledge base is open-ended in the sense that all possible actions are not known by the system.

The goal frame component provides an organizational structure for allowing knowledge to exist in modules called frames. A goal frame provides a representation for an object, situation, class or entity in terms of set of task parameter names and task parameter values for different task to be performed.

The object-rulebase component contains a set of pattern (premise) - action (consequent) assertional implications which model particular tasks by defining the logical relationships that exists between goals and subgoal pattern parameters for task.

The global database of state information contains a list of these parameters, along with their values and properties.

Finally the control scheme component acts as the interpreter for inference generation using a goal-driven inferential strategy and belief weighting factors to handle uncertainty in task parameter-values along with the problem solving logic.

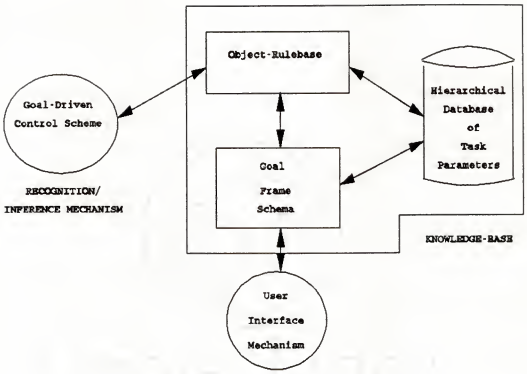


Figure 2.4 Proposed architecture for a knowledge-based decision support system for OA.

CHAPTER 3
AUTOSEC AND SYSTEM KNOWLEDGE FOR OFFICE AUTOMATION

3.1 Knowledge Categories for Office Automation

According to Croft and Leftkowitz (1984), in terms of task knowledge, not all aspects of every office tasks are, or can be, left to the discretion of individual office workers. Many office tasks contain what may be referred to as organizational knowledge aspects (OKAs) and individual knowledge aspects (IKAs). The OKAs of a task may define the overall task structure and usually impose some global requirements on the task. They represent policy set by the organization to which all individuals performing the task must adhere. The IKAs of a task, on the other hand, specify how a specific office worker accomplishes a specific task assigned to him/her. They represent the office worker's personal preferences and know-how for accomplishing a task. This mix of organization and individual aspect of a task allows the organization to maintain some control over tasks while still providing some freedom to the office workers performing the tasks. Our domain for task knowledge concentrates on the individual knowledge aspects (IKAs) of performing an office task for those tasks that are structured, such as meeting scheduling,

personnel decision functions and financial expenditure planning.

We classify office knowledge (incorporating administrative and professional within office) into four basic categories: 1) declarative knowledge; 2) procedural knowledge; 3) judgemental knowledge, and control knowledge (Tou, 1984; Tou, 1985). Declarative knowledge consists of definitions, statements, information patterns,etc., which are typically referred to as SURFACE knowledge. Procedural knowledge usually responds to questions concerning HOW, through the deployment of transformation rules. In the office domain, this refers to the sequence of subgoals that have to be accomplished in order to prove or achieve a goal. Procedural knowledge and declarative knowledge along with our inference strategy act to incorporate intelligence. Judgemental knowledge usually refers to questions concerning WHY. Both procedural knowledge and judgemental knowledge are referred to as DEEP knowledge. The declarative representation of office task knowledge also refers to the explicit representation of office activities along with their activation conditions. Finally, control knowledge is a function of the inference strategy which guides the overall operation for goal realization. Figure 3.1 shows a left to right categorical structure of knowledge categories.

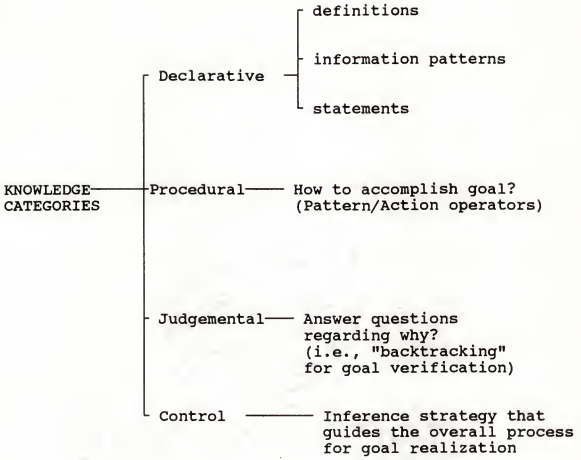


Figure 3.1 Task knowledge categorical structure

The nature of office task knowledge can be further characterized as being open-ended in a domain and organizational sense. The following subsections will briefly touch upon these two aspects of office knowledge characterization.

3.1.1 Evolutionary Domain Knowledge

It is not possible to describe, a priori, for all new and existing office work, the knowledge required to handle all domain situations. Office work occurs in a dynamic environment. A knowledge-based office support system (KBOSS) must be able to assimilate new information as it becomes available and to operate with incomplete, limited domain knowledge.

3.1.2 Evolutionary Organizational Knowledge

It is not possible to describe completely the organizational guidelines and constraints and the informal and formal relations among participants in office work. A knowledge-based office support system must be able to handle change and evolution in the organizational structure and operate with incomplete, limited organizational information. This kind of knowledge usually evolves much more slowly than domain knowledge. For example, new employees are hired and

added to the organizational structure less frequently than information from suppliers arrives to a purchasing department.

3.2 Knowledge base Structure and Representation

For office task decision support, the information to be encoded into the knowledge-base originates from descriptive statements that are often difficult or unnatural to represent by simple structures like arrays or sets of numbers. For example, tasks such as meeting scheduling, personnel decision functions, and financial planning require the capability for representing, retrieving and manipulating sets of statements. Figure 3.2 shows an abbreviated knowledge hierarchy for a meeting scheduling task.

Since a knowledge base is the encapsulation of human knowledge in the form of several data structures that can be processed by machine, we must employ structures that are capable of translating human experience and expertise into machine understandable information. Research in the knowledge representation area of artificial intelligence (AI) which initially came out of computer vision research has led to the formulation of several methods for knowledge representation by computer such as semantic networks, mathematical logic, procedural representations, scripts, productions, and frames described in Chapter 2.

The knowledge base is organized as a hierarchical collection of goal frames, Figure 3.3 shows an example goal

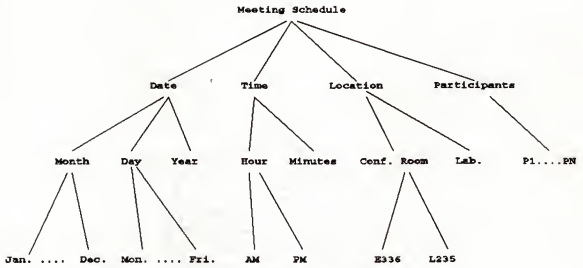


Figure 3.2 Hierarchical abbreviated portion of task parameters for a meeting scheduling task.

frame hierarchy. This structure acts to partition each task into a specific frame where each goal frame contains its own object-rules and goal parameters. Goal parameter determination represents the solution to the instantiated frames goal. Figure 3.4 shows the general knowledge base organization using goal frames. The following analogy is applicable in describing the knowledge organization.

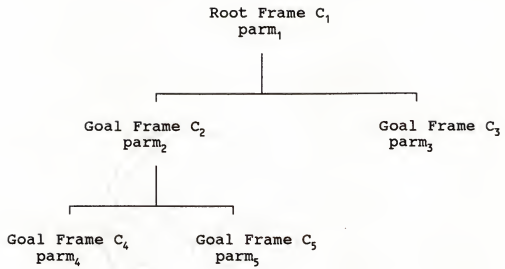
ENTITIES -> GOALS(goal parameters)

ATTRIBUTES -> SUBGOALS(task parameter values)

RELATIONSHIPS -> OBJECT-RULES

The above analogy holds since we bijectively map task parameters to task parameter values (subgoals), thus defining attributes of entities as subgoals of goals.

The organization and representation of office task in terms of task parameters is as follows: Goal Frame Schema: A goal frame, C_i, defines a set of goal and task parameters representing any aspect of the task problem. For example, in the domain of personnel decision-making based on an applicants qualifications, one goal and several task parameters may be identified and the associated value assignments to object-rules in the knowledge-base. Some identified goal parameters and task parameters are: a) position (Goal); b) experience (Subgoal); and c) discoveries (Subgoal). Corresponding to the main goal, a goal (object) frame schema is defined for storing facts relevant to the goal of a particular task. Goal frames,



Goal Frame C_2 = [Goal Frame C_1,($parm_1$,$parm_2$)]
Goal Frame C_3 = [Goal Frame C_1,($parm_1$,$parm_3$)]
Goal Frame C_4 = [Goal Frame C_2,C_1,($parm_1$,$parm_2$,$parm_4$)]
Goal Frame C_5 = [Goal Frame C_2,C_1,($parm_1$,$parm_2$,$parm_5$)]

Figure 3.3 Example goal frame hierarchy

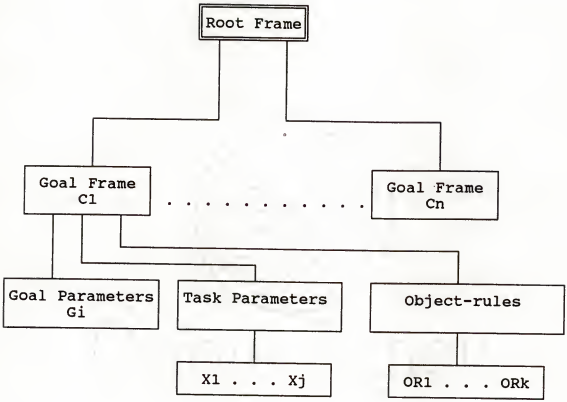


Figure 3.4 Knowledge-base Organization
Note: There exist a G1 Frame Schema expansion for each associated Goal Frame for task representation.

are utilized for their organizational effectiveness. Figure 3.5 shows two abbreviated goal frame schema for personnel assignment decisions and meeting scheduling. Task parameters: A set of task parameters are associated with a goal parameter. For example, the goal parameter "meeting schedule" may have several task parameters such as: a) participants; b) time; c) location; etc.

Hence a Goal Frame, C_i, has a set of task parameters $\{(X)_{ij}\}$ (i^{th} goal frame j^{th} task parameter) associated with it, where each task parameter represents an attribute of the particular goal parameter defined in C_i. Thus,

$$C_i = \{(X)_{ij} \mid 1 < j < k_i\}$$

where k_i, denotes the total number of task parameter attributes associated with C_i. Task Parameter-value: Each of the task parameters $\{(X)_{ij}\}$ can be mapped to a task parameter-value from a corresponding task parameter-value list S. Thus,

$$(X)_{ij} = (Q(S))_{ij}.$$

A task parameter value is associated with each task parameter to indicate the type of value to which the task parameter may be instantiated. As described earlier task parameters are attributes for task to be performed, and as such they are a part of an information expression utilized in decision-making operations. Goal parameters: An instantiated set of task parameters of a goal frame represents a set of task parameter-values in the global database.

C: Personnel Assignment		
X	Q(S)	OR id
Degree	------	----
Experience	------	----
Discovery	------	----
Academic	------	----
Industrial	------	----
Prev-Empl	------	----

(a)

C: Meeting Schedule		
X	Q(S)	OR id
participants	------	----
location	------	----
date	------	----
time	------	----

(b)

Figure 3.5 Abbreviated goal frame schema

(a) Example Goal Frame for Task of Personnel Assignment
(b) Example Goal Frame for Task of Meeting Scheduling

A goal parameter is a triplet consisting of a goal frame, a list of task parameter attributes, and task parameter-values to which the task parameters can be instantiated. An instantiated goal parameter represents the termination condition for goal realization.

$$(G)_{ijk} = \{(C)_i, (X)_{ij}, (S)_{ijk}\}$$

In the above $(S)_{ijk}$ is the k^{th} element of $(S)_{ij}$. For example, a) the instantiation of the task parameter "X_1 = discoveries" of the goal frame "C_1 = Personnel Assignment" to the task parameter-value "S_1 = yes" defines the goal parameter, "G_1 = The applicant for the position has made some discoveries"; and b) the instantiation of the task parameter "X_2 = location" in the goal frame "C_2 = Meeting Schedule" to the task parameter-value "G_2 = Center for Information Research" defines the fact, "The meeting will be held at the Center for Information Research."

3.2.1 Object Rulebase Organization

Task specific knowledge is organized as a set of object-rules contained in the task specific goal frames. As described previously, object-rules refer to the logical relationships which exist between task parameter-values contained in goal frames. For the finite set of task parameters X_i:

$$X_1, X_2, \ldots, X_n$$

the object-rulebase consists of a finite set of object-rules:

$$OR_1,\ OR_2,\ldots,\ OR_m$$

where each object-rule is of the form:

$$X_{i1},\ X_{i2},\ldots,\ X_{ip} \text{ ---> } X_j$$

where i1, i2,..., ip, j ϵ {1,2,...,n}, and has associated with it a coefficient belief weighting factor (BWF), $BWF(OR_i)$, of value between -100 and 100, that measures the strength of the object-rule; $BWF(OR_i) = 100$ corresponds to complete certainty. Calculations of belief weighting factors and how they are utilized are discussed in chapter 5.

The following is a simplified form of a object-rule in order to illustrate the concept:

Pattern ==> Action

where a Pattern is assumed to be comprised of task parameters in conjunctive or disjunctive form and an Action is the specific action required for a goal or termination condition. The antecedent or pattern component of an object-rule may consist of a single antecedent or a conjunction or disjunction of antecedents. The antecedents comprise the various subgoals needed in performing a task. For example, since the pattern components of object-rules are task parameters a sample rule is expressed as follows:

IF $X_1 = Q(S_1)$ and
$X_2 = Q(S_2)$ and
$X_3 = Q(S_3)$ and
.
.
THEN $G(X_i)$ = TRUE

The one-to-one and onto function Q maps task parameters to task parameter values, where the pattern elements $X_1 \ldots X_n$ are the task parameters needed for goal G to be realized. The action components may also consist of a conjunction or disjunction of consequents. This component indicates the action to be performed, for example the transfer of decision-path or the initiation of queries to the user. Moreover, in order to enhance the reasoning power, heuristic measures in the form of belief weighting factors may be attached to individual task parameters contained in patterns or actions. The use of uncertainty measures in the context of office problem solving primarily involves the assigning of values to task parameters whose values are uncertain. These situations typically arise when responding to queries to which the user is not certain. For example, for an office equipment purchasing task the user might not be sure of the exact price of the desired equipment, but can express a certain degree of certainty that it is within a certain range.

The query information pattern variables of task parameters contained in the pattern component of an object-rule are bound based on the corresponding goal frame task parameter-value. Pattern matching is performed using instantiated task parameter-values and object-rules in the knowledge base until a goal parameter G is realized.

3.2.2 Symbolic Pattern Matching in the Object Rulebase

A good way to specify a condition in an object-rule is by providing a pattern which the input should match if the condition is to be satisfied. Such a pattern in our case for office task decision support can be a very particular one containing a single task parameter or a general one which matches any finite list containing a goal parameter as a action component. The act of matching is a comparison of a query pattern expression with task parameter pattern expressions contained as object-rules in the knowledge base. The objective of this pattern matching is to find whether the query pattern has the form or elements required by the knowledge base pattern to achieve a goal.

Rather than concern ourselves with the isomorphism of list structures, we provide order control in the matching of query pattern elements in our task parameters-values to object-rule pattern elements. Order control of matching is accomplished through the following mechanisms: equality, "variable" constructs to match any element, match any sequence of elements, and match any element satisfying an asertional implication.

The pattern matching algorithm we ultilize is a unification algorithm that matches two patterns. This procedure is explicitly developed for our backward-chaining

reasoning strategy, since Lisp, our implementation language does not provide any pattern matching utility programs.

Let us begin by thinking in terms of matching a pattern and an ordinary expression. Because the ordinary expression is used to represent bits of knowledge about a real or imagined world, we call the ordinary expression a datum. For example, the following datums could indicate the available times for a meeting for two persons P1 and P2:

```
(Time P1 9:45am)          ;Datum examples
(Time P2 8:00am)
```

Unlike datums, patterns in AUTOSEC contain elements called task pattern variables. In order to make pattern variables easy to recognize, we encapsulate them in two-element lists whose first element is a ? and whose second element is the pattern variable. For example, the pattern variables x and y appear in the following patterns:

```
(Time (? x) 9:45am))       ;Pattern examples
(Time P2 (? y))
(Time (? x) (? y))
(Time (? x) (? x))
```

When a pattern contains no pattern variables, that pattern matches a datum only if the pattern is exactly the same datum, with each corresponding position occupied by the same task parameter. Thus the pattern, (Time P1 9:45am) successfully matches the datum (Time P1 9:45am), but (Time P1 9:45) fails to match (Time P1 10:00am).

Note that the correspondence of pattern-variable expressions to datum elements is remembered and pattern-

variable expressions are bound to datum elements. If the same pattern-variable expression appears more than once in a single pattern, only the initial appearance acts like a wild card; subsequent appearances act like the datum element corresponding to the first appearance.

The symbolic pattern matching procedure utilized in AUTOSEC works by first-rest recursion, taking patterns and datums apart using FIRST and REST until they are both reduced to the point where matching is easy. Thus, MATCH reduces the problem of matching (Time (? x) 9:45am) with (Time P1 9:45am) to the problem of matching Time with Time and ((? x) 9:45am) with (P1 9:45am). Then, MATCH reduces the problem of matching ((? x) 9:45am) with (P1 9:45am) to the problem of matching (? x) with P1 and (9:45am) with (9:45am). Finally, the problem of matching (9:45am) with (9:45am) is reduced to matching 9:45am with 9:45am and NIL and NIL.

MATCH keeps track of associations between pattern variables and datum elements using an association lists of bindings. When a pattern and datum match, the association list is returned, as in the following examples:

```
* (match '(time (? x) 9:45am)
          '(time P1 9:45))
((X P1))  ; association list

* (match '(time P1 (? y))
          '(time P1 9:45am))
((Y 9:45am)) ; association list

* (match '(time (? x) (?y))
          '(time P1 9:45am))
((Y 9:45am) (X P1) ; association list
```

If there are no pattern variables, the association list will be empty. MATCH returns the symbol FAIL when the pattern and datum do not match. The symbolic pattern matching procedure is constructed using four auxillary procedures for adding bindings to association lists, finding bindings in association lists, and for extracting the key and value from particular bindings.

First, we construct ADD-BINDING, a procedure that adds a binding to an association list, given a pattern-variable expression like (? X) and a value like P1 (a person's name). To bind is to reserve a place in memory for a task parameter value to be associated with a particular symbol. To construct ADD-BINDING, we use the progressive envelopment technique, starting with sample values for three variables, PATTERN-VARIABLE-EXPRESSION, DATUM, AND BINDINGS:

```
(setf pattern-variable-expression '(? x))
(setf datum 'P1)
(setf bindings '((y 9:45am))
```

First we extract the pattern-variable out of the pattern-variable-expression:

```
* (second pattern-variable-expression)
X
```

Next we combine the pattern variable with the datum:

```
* (list (second pattern-variable-expression)
        datum)
(X P1)
```

Next, we add this to the front of the association list:

```
* (cons (list (second pattern-variable-exp)
              datum)
        bindings)
((X P1) (Y 9:45am))
```

We encapsulate the previous constructs creating the ADD-BINDING, EXTRACT-VARIABLE, and MAKE-BINDING procedures:

```
(defun add-binding (pattern-variable-exp datum bindings)
   (cons (make-binding
           (extract-variable pattern-variable-expression)
           datum)
         bindings)))

(defun extract-variable (pattern-variable-expression)
  (second pattern-variable-expression))

(defun make-binding (variable datum)
   list variable datum))
```

The following is an example using the ADD-BINDING procedure:

```
* (add-binding '(? x) 'P1 '((y 9:45))
((X P1) (Y 9:45am))
```

The FIND-BINDING procedure is as follows:

```
(defun find-binding (pattern-variable-expression binding)
   (assoc (extract-variable pattern-variable-expression)))
```

The following is an example using the FIND-BINDING procedure:

```
* (find-binding '(? x) '((x P1) (y 9:45am))
(X P1)
```

The final two procedures for extracting keys and values are as follows:

```
(defun extract-key (binding)
  (first binding)

(defun extract-value (binding)
  (second binding))
```

The symbolic pattern matching procedure utilized in the object rulebase can now be given by adopting the problem reduction and the comment translation techniques, dividing the general problem into several more specific subproblems:

```
(defun match (p d &optional bindings)
  (cond
   (( and (atom p) (atom d))
```

```
;;See if P and D are the same
;;If so, return the value of BINDINGS
;;Otherwise, return FAIL
...)
((and (listp p) (eq '? (first p)))
;;See if the pattern variable is known
;;If it is, substitute its value and try again
;;Otherwise, add new binding
...)
((and (listp p) (listp d))
;;See if the first parts match producing new bindings
;;If they do not match, fail
;;If they do match, try the rest parts using the resulting
  bindings
...)
(t 'fail)))
;;Default Fail
```

3.2.3 Task Partitioning Criterion

The hierarchical goal frame structure of the knowledge base facilitates the partitioning of task object-rules into spaces based on the goal frame structure that is instantiated. An object-rule space is a set of task dependent object-rules. Thus, generated solutions are contextually dependent.

Given a set of goal frames, $\{C_1, C_2, \ldots, C_i\}$, a set of object-rules, $\{OR_1, OR_2, \ldots, OR_j\}$, and a set of task parameters $\{X_1, X_2, \ldots, X_k\}$, a task object-rule partitioning criteria exists that can be defined as a triplet consisting of the union of specific elements of the goal frames, goal object-rules, and task parameter sets. Thus, if **P'** is the set of all possible partitioning criteria then:

$$\mathbf{P'} = \{C_i\} \cup \{OR_j\} \cup \{X_k\}$$
$$= \{P_1', P_2', \ldots, P_i', \ldots, P_n'\}$$

where P_i' represents the partitioning criteria for the i^{th} goal object-rule space. The main advantage in partitioning the object-rule space lies in enabling the inference mechanism to focus its attention on a single object-rule space at a time. As a result the object-rules that are not relevant to the current context are eliminated and the search space is reduced.

CHAPTER 4
AUTOSEC KNOWLEDGE-BASED SYSTEM SEARCH AND CONTROL ISSUES

4.1 Distinguishing Between Search and Control

Problem solving systems can usually be described in terms of three main components. The first of these is a database, which describes both the current task-domain situation and the goal. The database can consists of a variety of different kinds of data structures including arrays, lists, sets of predicate calculus expressions, property list structures, and semantic networks. In theorem proving, for example, the current task-domain situation consists of assertions representing axioms, lemmas, and theorems already proved; the goal is an assertion representing the theorem to be proved. In information-retrieval applications, the current situation consists of a set of facts, and the goal is the query to be answered. In robot problem solving, a current situation is a world model consisting of statements describing the physical surroundings of the robot, and the goal is a description that is to be made true by a sequence of robot actions.

The second component of problem-solving systems is a set of operators that are used to manipulate the database. Some examples of operators include: in theorem proving, rules of inference such as modus ponens; in symbolic integration, rules

for simplifying the forms to be integrated, such as integration by parts or trigonometric substitution. Sometimes the set of operators consists of only a few general rules of inference that generate new assertions from existing ones. Usually it is more efficient to use a large number of very specialized operators that generate new assertions only from very specific existing ones.

The third component of a problem-solving system is a control strategy for deciding what to do next, in particular, what operator to apply and where to apply it. Sometimes control is highly centralized, in a separate control executive that decides how problem-solving resources should be expended. Sometimes control is diffusely spread among the operators themselves.

In problem-solving, it is often necessary to find paths through networks or tree structures. When knowledge-based programs deal with such problems, the result is two paths, one being a trace through the explored situations or places, the other being a trace through the methods that the knowledge-based program itself chooses to use. Figure 4.1 illustrates this. What happens depends jointly on the structure of the domain and on the structure of the problem solver.

In distinguishing search from control, if we concentrate on how a system works through a network of tree structures natural to a target domain, we are studying search. If we

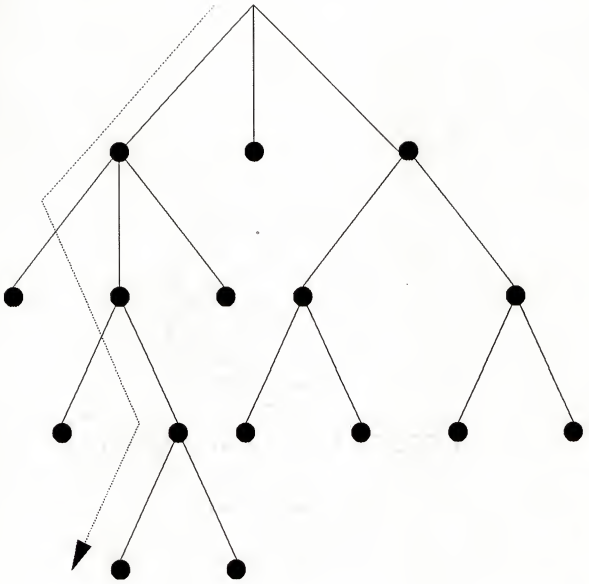


Figure 4.1 Search focuses on methods for exploring the tree structures that frequently describe problem domains. Control topics focus on how the problem solver chooses to shift attention among its subprocesses.

concentrate instead on how the system selects among its internal methods, we are studying control.

Clearly there is no boundary dividing search from control. The question is one of focus or point of view. One requires the other to make sense, just as hard work and brains are different, but are required jointly for successful scholarship. Moreover, some people prefer to regard control as that special case of search for which the search locus moves through a problem solver's individual methods.

There continues to be argrument about whether search is more important than control. Probably the extremist on both sides are wrong, both search strategy and control decisions can be simple, given good problem descriptions (Winston,1979).

4.1.1 Depth First Search

Given that one path is as good as any other, one agreeable idea is to pick some alternative at every node visited and work forward from that alternative. Other alternatives at the same level are ignored completely as long as there is any hope of reaching the destination using the original choice. This is the essence of the depth-first search idea. Using a convention that the alternatives are tried in a left-to-right order, the first action in working on the situation in Figure 4.2 would be a headlong dash to the bottom of the tree along left-most branches.

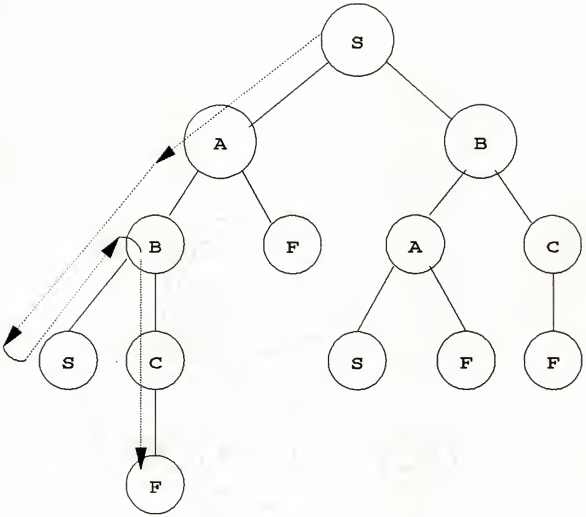


Figure 4.2 Depth-first search

But since this original set of decisions regarding alternatives fails to lead to the F node, the next step is to traverse backwards to the nearest node with an unexplored alternative. The last such node is B. The remaining alternative is better, bringing eventual success through C as shown. If the path through C proved not viable, then the process would move still further backwards through the tree seeking another viable decision point to move forward from. On reaching A, movement would go down again reaching the destination from there.

Depth-first search is an aggressive, but dangerous procedure. Imagine a more complicated tree with many more levels than in the simple tree illustrated. A process doing depth-first movement through such a tree is likely to slip past the level at which the F node appears and waste incredible energy in exhaustively exploring parts of the tree lower down. For such trees, depth-first search, while easily implemented, is the worst possible approach.

4.1.2 Breadth-First Search

When depth-first search is a poor idea, breadth-first movement may be useful. Breadth-first searches look for the destination among all nodes at a given level before using the branches descending from those nodes to push on. In the situation shown figure 4.3, B would be checked just after A. The process would then move on to the next level then discover

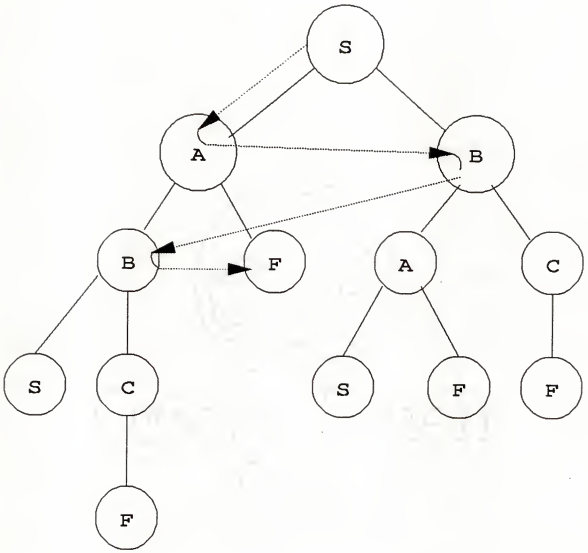


Figure 4.3 Breadth-first search

the F node just after noting that A leads to B as well. This will work even if the tree were infinite or effectively infinite. Although the breadth-first idea is careful and conservative, it can be wasteful. If all paths lead to the destination node at more or less the same depth, then breadth-first search works harder than depth-first search.

4.2 Office Task Problem Solving and AND/OR Trees

As opposed to the search strategies previously presented, we now consider a search idea that can be relevant to more general problem solving. In particular, we develop the AND/OR tree idea which descends from two obvious heuristics, namely, 1) try to convert a hard problem into one simpler equivalent, and 2) try to convert a hard problem into several simpler subproblems. Consider the first heuristic. There may be more than one simpler problem that is equivalent to the hard one. Moreover, the best way to solve any one of the possible simpler problems may be to convert it also. The natural result, then, is another tree structure in which each node represents a problem to solve. Such a tree is called an OR tree since at any node, branches lead to descendant problems which solve the parent problem. Solving any descendant problem is enough. More generally, solving any problem at the bottom of an OR tree solves the original problem as shown in Figure 4.4.

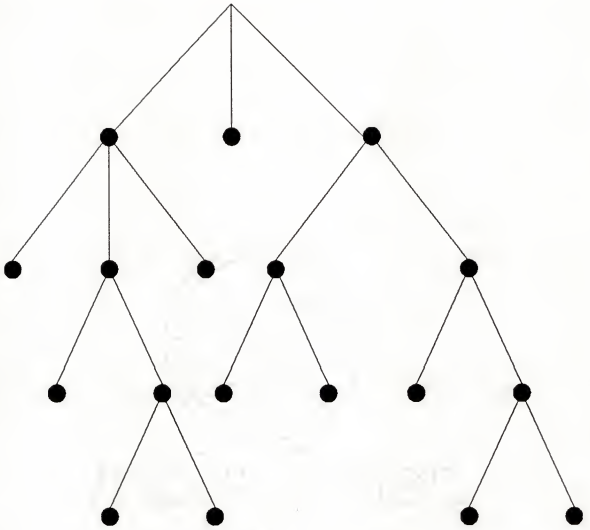


Figure 4.4 OR tree

The second heuristic implements the first because all the subproblems must be solved before the heuristic does any good. Repeated use of the idea also yields a tree structure, this time an AND tree, shown in Figure 4.5. By tradition, all the branches at AND nodes are tied together with an arc to emphasize that they help only when they all work together. Note that all the problems at the bottom of an AND tree must be satisfied before the original problem is solved.

One rarely sees a pure OR tree or a pure AND tree. Instead both types of nodes are freely intermixed, giving trees like the one shown in Figure 4.6. Of course it is possible to break up a problem into several different sets of subproblems suggesting a need to have nodes with mixed AND and OR qualities. For the sake of elegance, mixed nodes are avoided by inserting extra OR nodes into the tree. In summary, AND/OR trees are mixed collections of pure AND nodes, which show how a problem can be transformed into an equivalent set of subproblems, and pure OR nodes, which show how a problem can be transformed into any one of a set of equivalent problems. The AND/OR tree typically reveals that many different collections of problems at the bottom suffice to solve the original problem at the top. The structure of an AND/OR tree is often studied with interest because it reflects something of the nature of the particular problem at hand as well as the class of problems of which it is typical, in terms of decision processes.

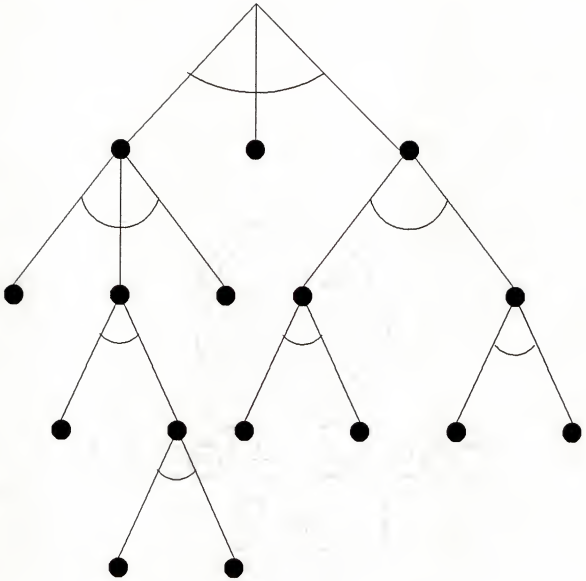


Figure 4.5 AND tree

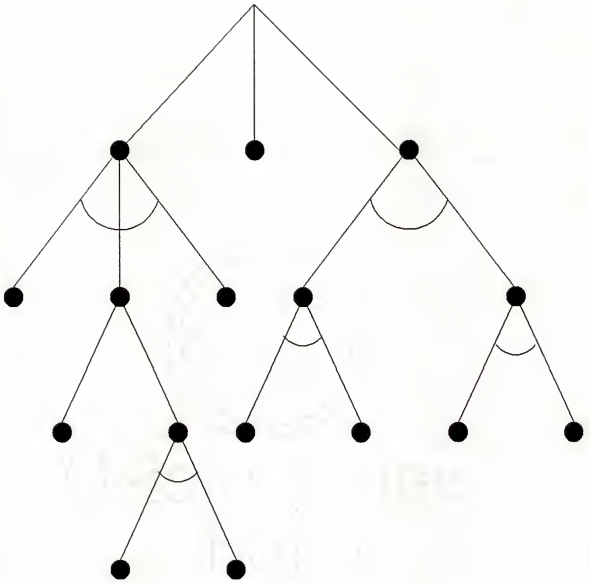


Figure 4.6 Mixed AND/OR tree

In AUTOSEC, we utilize the problem reduction technique, previously discussed in describing our office task model, that is, achieving the desired goal can be equated with finding an appropriate finite sequence of applications of available object-rules. Since the goal situation or the sequence that leads to it is of primary interest, the term search can always be understood, without misleading consequences in AUTOSEC, as referring to the search for an appropriate object-rule sequence.

Tree structures are commonly used in implementing control strategies for the search. For example, in a state-space representation, a tree may be used to represent the set of problem states produced by operator applications. In such a representation, the root node of the tree represents the initial problem situation or state. Each of the new states that can be produced from this initial state by the application of just one operator is represented by a successor node of the root node. Subsequent operator applications produce successors of these nodes, and so on. Each operator is represented by a directed arc of the tree. In general, the states are represented by a graph rather than by a tree, since there may be different paths from the root to any given node.

However, besides these ordinary trees and graphs, which are used for state space representation, there are also specialized ones called AND/OR graphs that are used with problem solving methods involving problem reduction which lie

at the heart of the AUTOSEC knowledge-based goal driven approach. AND/OR graphs provide a means for keeping track of which subgoals have been attempted and which combinations of subgoals are sufficient to achieve the original goal.

Let us now consider another simple example illustrating the process, which involves the utilization of an AND/OR tree construction.

In the search process, we only look at those object-rules in the knowledge base that have the desired goal in their consequences or action components; if one of the object-rules has all its patterns or antecedent components in the task parameter set we have succeeded, otherwise we take various antecedents in order as new goals (i.e. subgoals of the original goal) and start again. We are thus led to search a tree, which in simple cases, as in this subsequent example is finite. The way the recognition/inference mechanism goes about the task is determined by the form of certain object-rules, or by the way they are written.

Consider as an example the following and process, given the following object-rules:

OR_1: B^D^E==>F
OR_2: G^D==>A
OR_3: C^F==>A
OR_4: B==>X
OR_5: D==>E
OR_6: X^A==>H
OR_7: C==>D
OR_8: X^C==>A
OR_9: X^B==>D.

In addition to the above knowledge base, there must again be case-specific task parameters (ie. factual information pertaining to specific tasks). The above knowledge base might be accompanied by the following task parameters:

Task parameters: B, C

Now suppose for illustrative purposes, that the following query is made:

Goal H

meaning: Does H follow from the given task parameters, using the given object-rules? Using the above knowledge base and task parameters, the goal H occurs in the consequence of OR_6 only, so:

NEW GOALS (ie. SUBGOALS) X,A

OR_4 has X as consequence, so:

NEW GOALS (ie. SUBGOALS) A,B

and since B is in the given task parameter set we now have:

SUBGOAL A

Three object-rules (OR_2, OR_3, and OR_8) have A as a consequence, so the AND/OR tree to be searched has three branches:

Branch 1 OR_2: SUBGOALS G,D
G is not a consequence of any rule, so we are checked (unable to proceed down this path)

Branch 2 OR_3: SUBGOALS C,F
C is a given task parameter: new SUBGOAL F
OR_1: SUBGOALS B,D,E
B is a given task parameter: new SUBGOALS D,E
OR_7: new SUBGOALS C,E
C is a given task parameter: new

SUBGOAL E
OR_6: new SUBGOAL D
OR_7: new SUBGOAL C
C is a given task parameter: **success**

There is no need to search Branch 3. The process is shown diagrammatically in an AND\OR tree in Figure 4.7. Due to the nature of the goal-driven inference strategy, a consequent component of an object-rule is executed only once during a frame instantiation.

In the next section, we discuss the inadequacies of using a Bayesian inference and decision-making approach to handle uncertainty in knowledge base task parameters

4.3 Bayesian Inference and Decision Making

In this section, we review two approaches for inference and decision-making in knowledge-based systems. These approaches have primarily been used for diagnostic problems. We will discuss why these approaches are not applicable for decision support for office tasks.

4.3.1 Relative Frequency

For decision-making, probability theory has been used typically to provide a "likelihood" of one category versus another category. It provides a measure of how close features, x, are to a category. We would like to be able to learn this closeness measure either using expert knowledge or

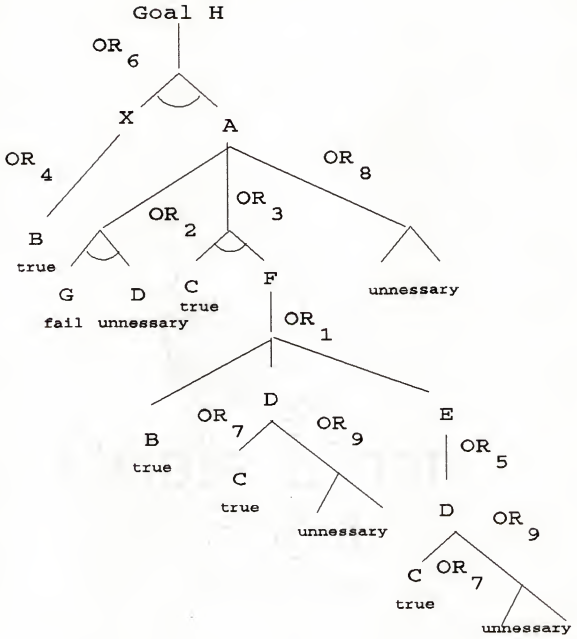


Figure 4.7 AND/OR tree example representation

hard data examples. Since the framework of probability theory has been used to embrace a likelihood or relative frequency or vote or experience, we can define experience, in the our approach, as "the process of encountering or understanding things as they occur in the course of time."

In order to incorporate experience in an intelligent system, an approach is needed which will assure that knowledge obtained for the target system is appropriate to solve a problem in the given domain of interest.

One method of obtaining experience is the recording of experience through experiments or trials to obtain estimates of those a priori and conditional probability density functions used in decision-making. We define the observation of only one goal category Ci, as

$$n_{(Ci)}$$

which is the relative number of times out of n times that goal category Ci is observed. We can represent this estimate of observing goal category space by

$$P(C_i) = n_{(Ci)}/n \qquad \text{eq. 4.1}$$

which describes the observation of goal category space as the frequency of goal category C_i. Equation (4.1) can be defined as a relative frequency estimate of the a priori probability of goal category C_i given a priori knowledge about a given goal category.

Extending this process to the observation of goal feature space F, $x \in F$, and the cross-product space (x,C_i), that is the interaction between task parameters and goal frame categories. We can define

$$P(x) = n_{(x)}/n \qquad \text{eq. 4.2}$$

as the observation of goal feature space, and

$$P(x,C_i) = n_{(x,Ci)}/n \qquad \text{eq. 4.3}$$

as the observation of cross-product space. We have in equation (4.2), $n_{(x)}$ can be defined as the number of times out of n times that a goal feature variable occurred, and $n_{(x,Ci)}$ in equation (4.3) the number of times the joint event (x,C_i) occurred. This joint event could be used to describe the relationship between an instantiated goal feature variable and the a priori goal frame category space. Applying the definition of conditional probability we have

$$\begin{aligned} P(x|C_i) &= P(x,C_i)/P(C_i) \qquad \text{eq. 4.4} \\ &= [n_{(x,Ci)}/n]/[n_{(Ci)}/n] \\ &= n_{(x,Ci)}/n_{(Ci)} \end{aligned}$$

which is the relative number of times goal feature variable x occurred with goal category C_i, compared with the number of times C_i is observed. This is typically called experience

(Patrick and Fattu, 1986). Experience results even if

$$n_{(x,Ci)} = \begin{array}{ll} 1 & x \text{ related to goal category } Ci \\ 0 & \text{otherwise} \end{array}$$

From the equations (4.1), (4.2), (4.3), and (4.4) we can formulate a relative frequency interpretation of Bayes theorem which results in the following equation

$$P(C_i|x) = P(x|C_i)P(C_i)/P(x) \qquad \text{eq. 4.5}$$
$$= (\,[\,(n_{(x,Ci)}/n_{(Ci)}) * (n_{(Ci)}/n)\,]/[\,(n_{(x)}/n)\,]\,)$$

where $P(C_i|x)$ refers to the posterior probability and * the multiplication operation. The major drawback in utilizing this approach in the office environment is the determination of what constitutes the hard data samples needed to obtain the a priori and conditional probability density functions, and the significant amount of data needed to obtain accurate estimates of the probabilities. In the next section, we discuss the expert knowledge approach and present the limitations of this approach when applied to automating office tasks.

4.3.2 Expert Knowledge

In the design strategy, we view the office domain as one that consists of the sort of real-world tasks that do not defy straightforward categorical classification. However, a

difficulty arises from the complexity and lack of "expert" knowledge for performing a task. However, all tasks have goals and purposes which can be utilized to construct goal task parameters or attributes for each goal activity object-rule set.

In the previous section, we interpreted Baye's rule from a relative frequency approach. Another interpretation uses the posterior probability $P(C_i|x)$, which describes the occurrence of goal frame category, C_i, based on the existence of evidence in the form of goal features, x. Thus, the posterior probability can be expressed as follows:

$$P(C_i|x) = P(x|C_i)P(C_i)/P(x) \qquad \text{eq. 4.5.1}$$

where P(x) is the probability of the occurrence of goal feature, x.

According to probability theory (Patrick and Fattu, 1986) the conditional probability of a set of goal feature variables, x, can be calculated as the algebraic sum of the conditional probabilities of the individual goal feature variables, x_j, provided that all task parameter variables exists such that they are mutually exclusive events in the goal parameter space, F. Thus,

$$P(x|C_i) = \sum_{x_j \epsilon F} P(x_j|C_i) \qquad \text{eq. 4.6}$$

where X_j denotes the $_j$th goal feature employed for goal realization. Since our goal frame categories were defined previously to be mutually exclusive, the goal feature variables for each office task description to be supported are events that are mutually exclusive to a given goal frame category. Thus, we conform to the criteria required for this approach. Using equation (4.2) and equation (4.6) we obtain

$$P(C_i|x) = \sum_{x_j \epsilon F} P(x_j|C_i)P(C_i)/P(x) \qquad \text{eq. 4.7}$$

since the denominator of the above is a constant for a goal feature variable, x, it may be discarded, simplifying equation (4.7) we obtain

$$P(C_i|x) = \sum_{x_j \epsilon F} P(x_j|C_i)P(C_i) \qquad \text{eq. 4.8}$$

From equation (4.8), we can utilize the relative magnitudes of the posterior probabilities as a "closesness or similarity" measure between end-user specified goal feature and a priori instantiated goal feature stored in the goal frame category office activity rule set (object-rule set) in the knowledge-base. This approach circumvents the hard data sample relative frequency interpretation problem, but creates another problem In equation (4.8), the conditional probability $P(x_j|C_i)$ and probability of occurrence of goal frame category, $P(C_i)$, will be difficult to obtain from end-user or expert knowledge due

to the nature of the office domain. Even though the Bayesian approach has been utilized for inference and decision-making in several knowledge-based systems, it is not applicable to our problem. Reasons behind these inadequacies and the method we employ for fuzzy logic to handle uncertain or inexact information in decision making will be described in Chapter 5.

CHAPTER 5
AUTOSEC SYSTEM INTEGRATION AND IMPLEMENTATION

5.1 Office Task Knowledge Processing

Once a task has been identified and stated, the first step is to decompose the task such that its many aspects are illustrated as simply as possible. For example, consider the task of a personnel director, whose job it is to decide whether or not a job applicant qualifies for a position, and if so, what position. The personnel director's job is to ask the right questions, drawing upon the knowledge he or she possesses, and to determine which position is best. Using this example, we now proceed to integrate the previous concepts in this dissertation into an implementable design framework for office task knowledge processing and decision support.

The diagram representation for this particular task can best be illustrated in an AND/OR decision tree. This is a very useful and effective type of diagram because it enables the visualization of all factors that must be considered in reaching a decision and also demonstrates how one consideration leads to others, which then still lead to others, and so on. Because many office task problems are

complex and difficult to conceptualize, we utilize decision trees to illustrate the problem clearly. Figure 5.1 is a decision tree for the previously described task of personnel job assignment. The flow diagram in this example consists of broken and solid line rectangular shapes, which define nodes connected by branches. The broken line reactangular shapes are decision nodes containing questions. The solid line rectangular shapes contain the goals of the diagram, which in AUTOSEC signify conclusions. The arrow lines designate the directional flow of the diagram. Each node has a numeric association for reference. The path taken from each node is determined by the response to the condition contained in the node. For example, node 5, shown in Figure 5.2, asks a question which has two possible answers and therefore two possible paths depending on the response, in this case what the applicant's school average is. If the average is less than a preset threshold, for example 3.5, path 1 will be taken. The applicant's average can be considered to be a task parameter variable X_i, and its task parameter value S_i, the intantiation of the task parameter. Therefore, each broken line rectangular node contains a task parameter and the paths are conditions upon the task parameter values of that task parameter. In constructing object-rules for this task domain, these conditions will become pattern components of an AUTOSEC object-rule. The solid line rectangles are goals or subgoals. For example, the rectangular box in Figure 5.3 would be the

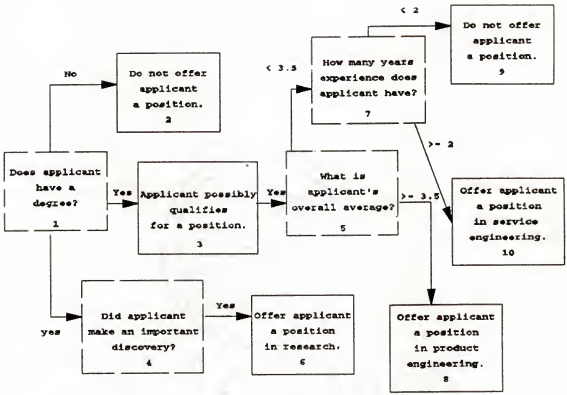


Figure 5.1 AND/OR decision tree for personnel assignment.

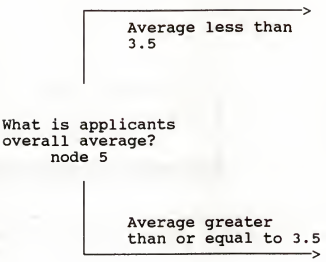


Figure 5.2 Decision node path determination.

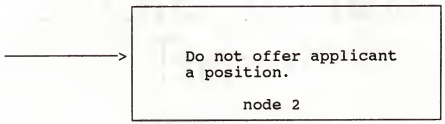


Figure 5.3 A goal or conclusion node.

answer to the problem statement "Should applicant be offered a position?" The goal would be to provide the answer. Likewise, the rectangular box in Figure 5.4 would be a goal of the problem statement, "Does applicant possibly qualify for a position?" However, it could also be a part of a path leading to another goal since it has a branch leaving its node. In that case, since that branch is not conditional, that is, it does not depend on any others, it is a subgoal of another goal. A subgoal is a pattern component of an object-rule.

Now that the task has been carefully and thoroughly diagrammed for task knowledge processing, object-rules which comprise the AUTOSEC knowledge base can be constructed.

5.2 Office Task Object-Rule Construction

As previously discussed in Chapter 3, AUTOSEC task object-rules are made up of two parts. The pattern or antecedent part is comprised of task parameters connected by disjunctive or conjunctive logical operators. The action or consequence part is evaluated only if the pattern part is true. The combination of linked decision nodes (broken line rectangular shapes) and goal nodes (single rectangular shapes) represents an object-rule. The pattern part contains all the decision nodes in the path leading to a goal node; each decision node in the path leading to a goal node contributes one task parameter to the pattern portion. The goal node itself forms the action portion. Figure 5.5 illustrates

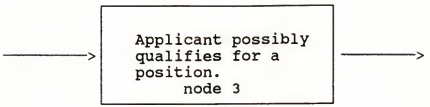


Figure 5.4 Example subgoal node.

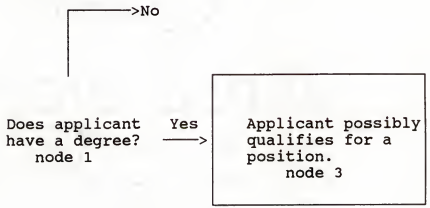


Figure 5.5 Object-rule path conversion

an example. In Figure 5.5, to detemine if the applicant qualifies for a position, node 3 is the goal node. The only path leading to this goal contains decision node 1, "Does applicant have a degree?" the object-rule that this path generates contains a single task parameter (degree):

Pattern: IF applicant has a (x_1:degree = $Q(s_1)$:yes)
Action: THEN applicant possibly qualifies for a (G_1:position = $Q(s_1)$:yes)

The clause "applicant has a degree" can have one or two values, namely yes or no. Therefore, "applicant has a degree" is a task parameter variable. In a computer implementation, we subsequently assign each node a short task parameter name. Table 5.1 lists all task parameter names contained in the nodes of the diagram in Figure 5.1, along with the task parameters full meaning and the corresponding numeric association of each node in which the task parameters can be found. These abbreviations simplify the creation of object-rules in the AUTOSEC knowledge base.

The AUTOSEC knowledge base object-rule generating technique for generating all the object-rules for each of the possible goals is performed using the following algorithm:

STEP 1: Choose a goal node of the decision tree.

STEP 2: Choose a decision node with a connecting branch to the left of the node chosen in STEP 1.

STEP 3: Continue STEP 2 until either there are no more nodes to the left or until a goal node is reached.

Task Parameter Variable Name	Meaning	Node(s)
DEGREE	Does applicant have a degree?	1
RESEARCH	Did applicant make an important discovery?	4
EXPERIENCE	How many years experience does applicant have?	7
AVERAGE	What is applicant's overall school average?	5
POSITION	What position should be offered to applicant?	2,6,8, 9,10
QUALIFY	Applicant possibly qualifies for a position.	3

Table 5.1

STEP 4: If a goal node is reached, record it and stop. If there are no more nodes stop.

In the object-rule generating algorithm described above, each decision node is a task parameter component of the pattern portion of an object-rule. All connected task parameters comprise conjunctive or disjunctive patterns, with the goal node comprising the action portion of an object-rule.

As an example of the AUTOSEC object-rule generation technique, consider the path shown in Figure 5.6. Following the steps previously outlined, the following listing is obtained using the node numeric associations:

Goal node: $G_1 = 6$
Path: $G_1 = 6$, $x_1 = 4$, $x_2 = 1$

Since we utilize a goal-driven inference strategy akin to backward chaining, described in a subsequent section, we start with the goal node and move backward through the decision tree.

From the above listing and the task parameter name table 5.1, the following object-rule is generated:

Pattern: IF $x_1 = Q(s_1)$ ^ $x_2 = Q(s_1)$
Action: THEN $G_1 = Q(s_2)$

where, x_1 = DEGREE
x_2 = DISCOVERY
G_1 = POSITION
$Q(s_1)$ = YES
$Q(s_2)$ = RESEARCH

That is, if the applicant has a degree and has made an important discovery, he or she should be offered a position as a researcher.

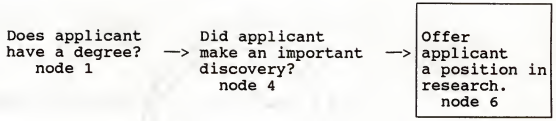


Figure 5.6 A sample path determination

The priciples in this section have been utilized to construct the AUTOSEC knowledge base for several office task domains. In the next section, we discuss the goal-driven inference strategy which is the control strategy that governs the processing of the knowledge base.

5.3 AUTOSEC Goal-driven Inference Strategy

When the human mind sets out to solve even the simplest problem, it has a vast store of information on which to draw in determining the proper course of action. If we do indeed have so much information to process all the time, how does the mind extract the right set of rules quickly to fit each particular situation?

There exist a more sophisticated system which guides the selection of a proper response to a specific situation. This process is called knowledge pruning. As it name suggests, pruning elminates pathways of thought that are not relevant to the immediate objective of reaching a goal. Just as one might cut off the lower branches of a tree to help its overall growth, so the pruning mechanism in our brains cuts us off from any facts and rules that won't lead us to a goal. When the mind is confronted with a situation, the pruning mechanism guides the thought processes by focusing only on those rules that are pertinent to solving the immediate problem.

An overview of how pruning helps the brain to distinguish one set of rules from another is shown in Figure 5.6. In this case pruning helps us decide whether to take path A or path B. It does this with an IF-THEN test of a condition. If the condition is A, it takes path A; if B, it takes path B.

Pruning can be likened to the task of an office manager who delegates tasks according to a set of priorities. The secretary, the word processor, the clerk, and everyone else on the staff have their assigned jobs to perform. But when everyone is called upon to help complete a particular project quickly, someone must decide not merely what everyone else has to do, but when they must do it. Without such decisions being made, some personnel might sit idle, waiting for others to complete related tasks, while some might waste precious time performing tasks that can wait until later. A lot of work might get done, but all that good energy is wasted if, in the end result, the goal of putting the project together properly isn't met.

Without pruning, our brains would be crippled in much the same way the office would without the manager. We would have to process every rule in our brains for every single problem we had to solve.

Everything we need to know to make decisions may be present in our brains all the time, for instance, we may know the capable members of the staff who are all doing what they are supposed to be doing, but unless the right information is

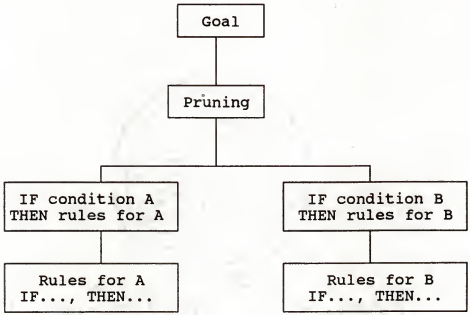


Figure 5.6 Pruning in action

pulled out just at the right time, all our knowledge is wasted. Pruning provides a vital order to our thoughts, without which life would not merely be slowed down, as in the office, but would be virtually impossible.

The inference generation method we employ is one based on consequent rules using a goal resolution stack, and a backward-chaining method. Consequent rules are the coded form (object-rule action component) of hypothesis that are tested in order to prove the goal parameter contained in the AUTOSEC goal frame knowledge base component. The strategy we employ first determines the goal parameter(s) that constitute the goal, by searching the action components of object-rules for the appropriate goal parameter for a given task. Object-rules that determine the task parameter value(s) associated with a given goal parameter are then tested. Consequent rules, employ a search method called backward-chaining which focuses on finding an object-rule to provide a necessary goal parameter value, i.e., it starts with a goal, G, and only applies those object-rules needed to establish subgoals to prove G. We begin with a specified goal (task) which is supported by task parameters (declarative knowledge). Object-rules are organized to provide observations in terms of information that is needed to carry out a particular task. The backward-chaining rule linking process creates a pattern using the search strategy that eliminates the firing of those object-rules not germane to achieving the desired goal

(essentially this process works as a pruning mechanism). To illustrate an abstract example, let us denote object-rules in goal frames C_i, as OR_j where $_j$ is the $_j$th object-rule of the object rulebase. Let G_i denote the $_i$th goal parameter for solution to a goal, and X_k denote the $_k$th subgoal (task parameter) required for goal realization. Figure 5.7 presents a simple but illustrative example backward-chaining scenario for subgoal formulation to goal realization.

The following simplified procedure describes the overall process. In step 1, the AUTOSEC knowledge base tries to establish (if it can) that goal parameter G_{10} exists by searching the database for object-rules that contain goal parameter G_{10}, and if the search fails, it searches for subgoals (task parameters) which conclude goal parameter G_{10}, that is have G_{10} as a consequence. It finds the object-rule $X_7 \wedge X_2 \Longrightarrow G_{10}$, and decides that it must establish subgoals (task parameters) X_7 and X_2 in order to conclude goal parameter G_{10}. In step 2, the knowledge base tries to establish task parameter X_7, first checking the database and finding an object-rule that concludes subgoal (task parameter) X_7. From this object-rule, $X_4 \wedge X_6 \Longrightarrow X_7$, the knowledge base reasoning strategy decides that it must establish $X_4 \wedge X_6$ to conclude task parameter X_7, since they are subgoals.

In steps 3 through 5, the knowledge base finds that task parameter X_4 is contained in the database, but decides that it must establish X_5 before it can conclude X_6, since X_5 is a

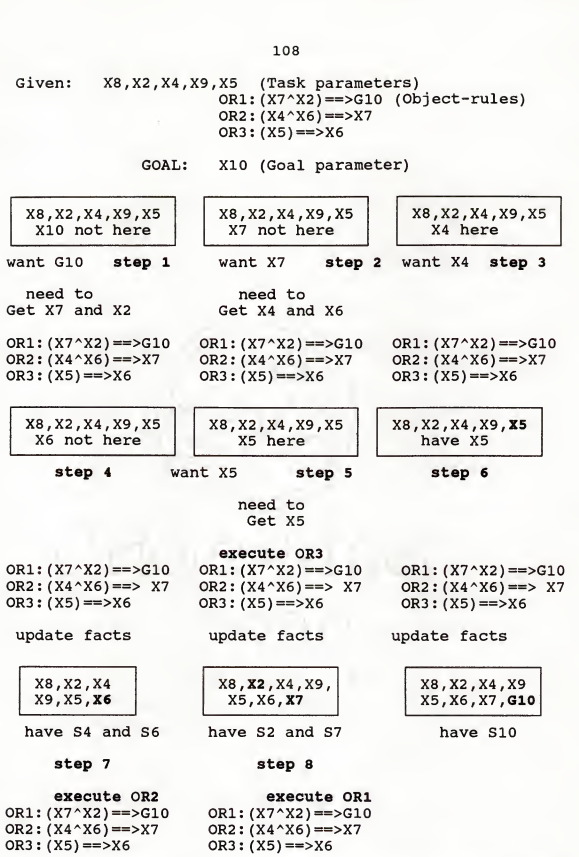


Figure 5.7 Example of subgoal formulation to goal realization using a goal-driven inference strategy

since X_5 is a subgoal, it then finds X_5 in the database.

In steps 6 through 8, the knowledge base control strategy executes the third object-rule, OR_3:(X_5) ==> X_6, to establish X_6, then executes the second object-rule, OR_2: (X_4 ^ X_6) ==> X_7, and finally executes the first object-rule, OR_1:(X_7 ^ X_2) ==> G_{10}. Thus, the pattern created during this backward-directed inferencing process is as follows:

$$G_{10} = (X_5 \text{ ^ } X_6 \text{ ^ } X_4 \text{ ^ } X_7 \text{ ^ } X_2)$$

When the value of a task parameter is needed, AUTOSEC determines the object-rule that might set the value of the task parameter during the previously described backward-chaining process.

One of the main advantages of this process is the automatic pruning of the search space that is accomplished by invoking only those object-rules that will add in accomplishing a desired goal. Another advantage of applying a goal-driven strategy for inference lies in its efficiency of practically doing away with combinatorial problems for small task knowledge bases. Particularly, for task comprised of limited domain knowledge. If an object-rule can provide the task parameters value, tests are performed on the object-rule pattern or antecedent to see if it is successful. At this point one of three situations can occur: 1) If the pattern or antecedent statement is successful, the action or consequence statement of the object-rule is performed, which sets the value of one or more task parameters; 2) If a task parameter

in pattern or antecedent statement has no value when it checks the object-rule, a search is performed for a object-rule to conclude the unknown task parameter value. AUTOSEC sees this task parameter as a subgoal and tests other object-rules that set the subgoal in their pattern or antecedent statement; 3) If there are no more object-rules left to determine the value a task parameter requires in a particular pattern or antecedent statement during interactive consultation, the user will be prompted for the task parameter value.

5.4 Data Structures for Office Task Decision Support

Often distinguished from the state-space representation of problems is the technique we employ called problem reduction representation. In the problem reduction approach we use for office task decision support, the principle data structures are problem descriptions or goals. An initial problem description is given and is solved by a sequence of transformations that ultimately change it into a set of subproblems whose solutions are immediate. The transformations permitted are defined as operators. An operator may change a single problem into several subproblems; to solve the former, all the subproblems must be solved. In addition, several operators may be applicable to a single problem, or the same operator may be applicable in several different ways. In this case, it suffices to solve the

subproblems produced by any one of the operator applications. A problem whose solution is immediate is called a primitive problem. Thus the problem representation using problem reduction in AUTOSEC is defined by a triple consisting of 1) an initial problem description (goal frame), 2) a set of operators for transforming problems to subproblems (object-rules) and 3) a set of primitive task descriptions (task parameters). Reasoning proceeds backward from the initial goal.

Using the knowledge base of object-rules we can derive several structures, that add in the incorporation of intelligence for task problem solving. Since these structures are derived directly from the knowledge base they serve an indispensable purpose.

The first structure is the goal parameter list. This structure lists all the possible goals of the knowledge base in sequential order. These goals correspond to the termination condition or objective of tasks. The goal parameter list contains three items: 1) object-rule id; 2) the goal parameter associated with the object-rule id; and 3) the task parameters (subgoals) that lead to goal realization. This data structure is as follows:

(OR ID)	G: goal-parm	X: task-parm

There is one entry in the goal parameter list for every object-rule in the knowledge base. The set of subgoals

contained in the goal parameter list are implicit, since we employ a backward directed inference strategy (i.e. by definition every goal parameter is defined by a finite and unique set of subgoals).

The second data structure is the task parameter list. This structure contains two items: 1) the task parameter name for each variable contained in the antecedent component of a object-rule; and 2) a valuation item that tells whether or not a task parameter has been instantiated. This data structure is shown as follows:

X1	NI	Q(S):VAL
X2	NI	Q(S):VAL
.	.	.
Xi	.	.
.	.	.
Xn	NI	Q(S):VAL

A task parameter can appear only once in this list independent of how many times it appears as a subgoal in various pattern components of object-rules. Also, we require the constraint that goal parameters cannot be defined as task parameters, since goal parameters terminate the solution process. The instantiated column of the task parameter list is always initialized to not instantiated (NI). It is appropriately modified as each task parameter is set to a specific value.

Before an object-rule in the knowledge base can be fired, all subgoals in the pattern component of an object-rule must first be instantiated. After a value has been assigned to a task parameter, we can set the instantiated column of the task parameter list to (I) along with the variable value. We are then in a position to match and/or compare the task parameters with subgoals of any antecedent component of an object-rule containing the the task parameter.

The third data structure is the subgoal variable list. Since pattern or antecedent statements can contain one or more task parameters connected by AND,OR, or NOT logical connectives, we assemble a list of subgoal variables for each antecedent statement of each object-rule. First we check the goal parameter list to see if each task parameter contained in the list has been assigned a value. We then instantiate those task parameters that have not and execute the object-rule in the knowledge base satisfying the implicational constraints. The data structure containing the list of task parameters in terms of subgoals is referred to as the subgoal variable list.

Since each goal parameter is comprised of a finite set of subgoals, we allocate the same number of memory locations for each object-rule. By prefixing each object-rule with an id corresponding to an array location, we can apply a simple hashing function to determine the location of each object-rule. To accomplish this process we use the following multiplication formula:

$$R_n = \{[(OR_A)*(OR_\#/OR_i)-1]+1\}$$

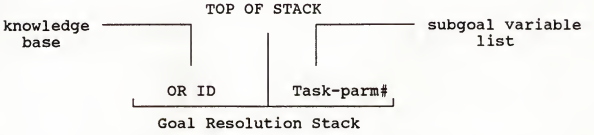
where

R_n - row ID of array index
OR_A - number of array memory locations allocated for each object-rule
$OR_\#$ - object-rule number prefix
OR_i - object-rule increment

The data structures we use for office task automation discussed so far, correspond to thought processes one might utilize when trying to solve a problem. First, we consider all possible pathways that would lead to a goal (goal parameter list). Then, we map out conditions that constitute those pathways (task parameter list and subgoal variable list). Because task parameter values often apply to more than one possible goal in a given situation, the data structures presented here enable us to manage this information quickly without continuous step repetition.

The fourth and last data structure we derive from the task knowledge base is the consequent or goal resolution stack. The goal resolution stack is the central structure that ties together all of the other structures discussed thus far. The goal resolution stack tells which object-rules contain the goal we are trying to reach, along with which task parameter is being examined for instantiation. This structure is a Last In/First Out (LIFO) stack that contains to items: 1) the object-rule ID containing the goal parameter; and 2) the subgoal variable number (i.e. task parameter number) in the

antecedent component of the object-rule. This structure is as follows:



The goal resolution stack is the structure we employ for keeping track of which object-rule and which subgoal within an object-rule is being verified.

5.5 AUTOSEC System Architecture

The system architecture for a knowledge-based system for applications for office task decision support is shown in Figure 5.8 This hybrid architecture is a variation of the rule-based and pattern-directed approaches discussed previously. This particular architecture utilizes the various data structures previously describe. The major component of this architecture is the knowledge base structure. The knowledge base consists of three major subcomponents, namely goal frame structures, global database of task facts for state information, and a set of object-rules describing the logical relationships between task parameters. Associated with these three subcomponents are three lists having different functions. The task parameter list contains the definitional characteristics of a task in terms of primitive information

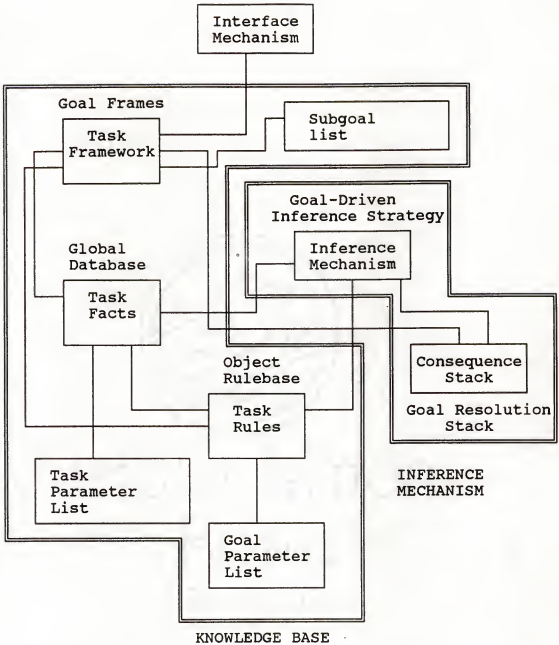


Fig. 5.8 Proposed AUTOSEC system architecture

atoms. The goal parameter list contains a list of goals of tasks in the knowledge base and the subgoal list is a valuation list containing task parameters-values which are mapped to goal frame task parameters.

The following algorithm implements the the AUTOSEC goal-driven inference strategy using the previously described data structures contained in the AUTOSEC architecture:

STEP 1: Identify the goal parameter.

STEP 2: Search the goal parameter list for the first instance of the goal parameter's name. If found, place the object-rule on the goal resolution stack using the object-rule ID number and a (1) to represent the the clause number of the task parameter.

STEP 3: Instantiate the IF clause (i.e., each subgoal condition variable of the pattern expression).

STEP 4: If one of the IF clause task parameter variables is not instantiated, as indicated by the task parameter list, and is not a goal parameter, that is not on the goal parameter list, query user to enter a value.

STEP 5: If one of the clauses is a goal parameter, place the goal parameter's object-rule number on top of the goal resolution stack and go back to STEP 3.

STEP 6: If the statement on top of the goal resolution stack cannot be instantiated using the present IF-THEN pattern-action statement, remove the unit from the top of the goal resolution stack and search the goal parameter list for another instance of that goal parameter's name.

STEP 7: If such a statement is found, go back to STEP 3.

STEP 8: If there are no more goal parameters left on the goal resolution stack with that name, the object-rule for the previous conclusion or goal is false. If there is no previous conclusion, then notify the user that an answer cannot be found. If there is a previous conclusion, go back to STEP 6.

STEP 9: If the object-rule on top of the goal resolution stack can be instantiated, remove it from the stack. If another goal parameter is underneath, increment the task parameter clause number, and for the remaining clauses go back to STEP 3. If no other goal parameter is underneath, the query has been answered and the user can come to a conclusion.

5.6 Uncertainty and Knowledge base Task Parameters

Task parameter information may be incomplete, erroneous, or inconsistent. The incorporation of distributed problem solving strategies in knowledge-based systems for decision support may necessarily lead to incomplete local views of an overall problem. Different workers may have different ways of describing and performing equivalent work. The knowledge of all agents (persons who perform task) taken together may be perpertually inconsistent. Information from other participants of the environment may not be correct and reliable. Therefore, a knowledge-based system for office decision support must be able to operate with uncertain knowledge.

In order to incorporate information based on uncertainty in our approach, we introduce what we term as "level of confidence" information. We employ belief weighting factors as opposed to a probabilistic framework for inexact reasoning for office task activities since they are not applicable to a probabilistic framework because of the following reasons: 1) It is often difficult to collect all the a priori conditional and joint probabilities required for some tasks. This would

require accumulating a great mass of data; 2) It is very difficult to modify the knowledge base of a Bayesian intelligent system because of the large number interactions between the various components of the knowledge base (knowing for example that the probabilities of all possible outcomes must sum to one). A problem is encountered when adding new data to the knowledge base, in that new entries might have a negative causal effect on the existing data which might require modication. 3) Evaluating Bayes formula in a complex domain such as the office environment for some office task activities requires a significant amount of computing, since so many probabilities must be considered. Some of which may contribute very little to the answer, so such detailed computation may be a waste of effort (Rich, 1983).

To handle task involving fuzzy logic we employ belief weighting factors, which are numeric values that indicate a measure of confidence in the value of a task parameter. Belief weighting factors are used to enable the object-rule structure and the expected task parameter value to comprehend uncertainty. These situations include those which involve the prediction of future events, guesses, and those which entail the subjective rating of a quantity.

Specification of a belief weighting factor for a task parameter value is accomplished by assigning the task parameter a numeric value between -100 and 100. This numeric value is assigned in the consequence statement of a object-

rule in the knowledge base associated with a given goal frame a priori or during problem solving. The range of belief weighting factors from -100 through 100 is subdivided into several ranges. Each range denotes a particular measure of belief. The extreme values are:

100 - absolutely confident that a task parameter value is true

-100 - absolutely confident that a task parameter value is not true

If the antecedent statement of a object-rule consists of one or more predicate task parameters that are combined by means of AND and OR logical connectives, and the clause is true, the task parameters have a belief weighting factor associated with it from predicate functions which return the belief weighting factor associated with the task parameter tested.

The belief weighting factor (BWF) associated with a pattern (antecedent) expression in the knowledge base is calculated as follows: If the pattern expression X_i (task parameters) are combined by means of the AND operator, each X_i must be true in order for the antecedent statement to be true. The belief weighting factor of the antecedent statement is the minimum belief weighting factor of the X_i. Any BWF greater than twenty that is attached to a task parameter is true.

pattern expression: $Q(X_1)$ AND $Q(X_2)$----> TRUE

$BWF_{and} = MIN \{Q(X_1), Q(X_2)\}$

If the X_i's in the pattern expression are combined by means of the OR operator, and the pattern expression is true, the belief weighting factor of the pattern expression is the maximum belief weighting factor of the X_i's.

pattern expression: $Q(X_1)$ OR $Q(X_2)$ ----> TRUE

$$BWF_{or} = MAX \{Q(X_1), Q(X_2)\}$$

When the action or actions stated in the consequence expression of an object-rule are taken, a new belief weighting factor for the task parameter-value is calculated. We combine belief weighting factors to produce new belief weighting factors for the task parameter-values. If the value of the task parameter being concluded has a previous belief weighting factor association with it, the new belief weighting factor will be calculated according to one of the following equations. These computations are based on integer arithmetic and will produce a rounded result. In these equations BWF(previous) or BWFp and BWF(new) or BWFn are defined as follows:

BWF(previous) = BWF associated with the task parameter value prior to the consequence statement statement being triggered

BWF(new) = [(BWF of pattern expression * BWF of conclusion function)+50]/100

If BWF_p and BWF_n are both positive or zero, the following equation is used:

$$BWF = BWF_p+[(BWF_n)*(100-BWF_p)+50]/100 \quad \text{eq. 5.1}$$

If BWF_p and BWF_n are both negative, the following equation is used:

$$BWF = BWF_p + [(BWF_n) * (100 + BWF_p) - 50]/100 \quad \text{eq. 5.2}$$

If the product of BWF_p and BWF_n is negative and the sum of BWF_p and BWF_n is positive, the following equation is used:

$$BWF = (BWF_p + BWF_n) * [100 + (100 + MIN/2)]/(100 - MIN) \quad \text{eq. 5.3}$$

If the product of BWF_p and BWF_n is negative and the sum of BWF_p and BWF_n is negative, the following equation is used:

$$BWF = (BWF_p + BWF_n) * [100 - (100 - MIN/2)]/(100 - MIN) \quad \text{eq. 5.4}$$

where, MIN = minimum of $|BWF_p, BWF_n|$ (TI, 1986).

Let us consider another example. The following object-rule from a sample knowledge base for buying or leasing an asset for an office is given; in testing OR005 let us assume the following task parameters have previous values and belief weighting factors as indicated.

Pattern Expression Task Parameters:

IF parameters:

```
LESSE-CREDIT          FAIR (70)
LENDER-CHECKS         YES  (50)
LESSE-CASH            FAIR (60)
CASH-RESERVE-NEEDED   YES  (90)
```

Consequent action:

THEN Goal Parameter:

PRESERVES-CASH (40)

OR005

```
IF: (pattern)   LESSE-CREDIT = FAIR AND
                LENDER-CHECKS AND
                LESSE-CASH = FAIR AND
                CASH-RESERVE-NEEDED
```

THEN: (action) PRESERVES-CASH BWF 90 AND PRINT "Since your lender checks on outstanding leases when making new loans, your organizations credit rating will be affected regardless of whether you buy or lease the asset. However, your business needs larger than average cash reserves which a down payments on a loan would deplete. Therefore, a lower front end cost of a lease will maintain your cash reserves."

Each clause in the IF statement shown in this example is true because each task parameter has a value and associated belief weighting factor greater than 20. Since the clauses are combined by means of the AND function, the weighting factor of the IF statement will be the minimum weighting factor of all the clauses in the IF statement. Therefore, the weighting factor of the IF statement is 50. Combine this 50 with the 90 weighting factor by using the equation for BWF(new):

$$BWF(new) = (50 * 90 + 50)/100 = 45$$

Since both BWF(new) and the previous belief weighting factor for PRESERVES-CASH (40) are positive, equation (5.1) is used to compute the new belief weighting factor for PRESERVES-CASH.

$$\begin{aligned} BWF &= 40 + ([45 * (100 - 40) + 50]/100) \\ &= 40 + 27 \\ &= 67 \end{aligned}$$

PRESERVES-CASH still has a value of YES but additional evidence increased the belief weighting factor to 67.

During interactive consultation, belief weighting factors are specified for a task parameter value by assigning it a numeric value between -100 and 100 as described previously. This numeric value is assigned in the THEN or action portion

of an object-rule in the knowledge base. These factors can also be updated during consultation.

When encoding the THEN portion of an object-rule, contributions can be made to the belief weighting factor assigned to a goal parameter value. However, the contribution to the goal parameter value's belief weighting factor is only partial, because of the manner in which our approach calculates the belief weighting factors. When the conditions of the IF statement of an object-rule are met, the appropriate belief weighting factors are combined, according to the previously given equations in the preceding paragraphs, and the result is assigned to the goal parameter value. These belief weighting factors are those considered when the belief weighting factor for the goal parameter is calculated.

The following belief weighting factors are combined to produce a new belief weighting factor for the knowledge base task parameter value: (1) the BWF of the IF statement; (2) the BWF of the THEN statement; (3) the task parameter being assigned a value that has not had one assigned before, and (4) the task parameter being assigned a value that has a previous value with an associated BWF of zero.

5.7 Experimental Office Task Domain Application

The AUTOSEC knowledge-based goal driven model for office task decision support, proposed in this dissertation, has been

successfully implemented, using the LISP (list processing) language on an IBM PS/2 model 30 personal computer, for three task applications: 1. Financial expenditure planning; 2. Meeting scheduling; and Personnel assignment decisions.

Samples of the task parameters, object-rules and and goal frames which make up the AUTOSEC knowledge bases for the implemented task are included in the appendix. The following is an example consultation with the AUTOSEC office asset financial expenditure decision support system for an office asset purchase versus lease decision-making application:

```
                        AUTOSEC:

A KNOWLEDGE-BASED GOAL-DRIVEN DECISION SUPPORT SYSTEM
                    FOR OFFICE AUTOMATION

** End - RETURN/ENTER to continue
```

```
TASK DOMAIN GOAL FRAMES:
> FINANCE
  PERSONEL
  SCHEDULE




1. Use the arrow keys or first letter of item to position
   the cursor.
2. Press RETURN/ENTER to continue.
```

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

```
How would you describe your office's current credit
status in terms of the following ratings?

          Yes
  ..........  GOOD
  .....>.....  FAIR
  ..........  POOR




1. Use an arrow key to indicate your degree of certainty
2. To select only one item, with 100% certainty, press
   CRTL-right arrow.
3. After making selections, press RETURN/ENTER to continue
```

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

Describe your office's access to cash reserves in terms of the following ratings?

```
        Yes
 ..... GOOD
 ...>..... FAIR
 ..... POOR
```

1. Use an arrow key to indicate your degree of certainty.
2. To select only one item with 100% certainty, press CRTL-right arrow.
3. After making selections, press RETURN/ENTER to continue

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

In your office, do you need to maintain larger than average cash reserves to take advantage of unexpected opportunities?

```
    YES
 >  NO
```

1. Enter a positive number.
2. Press RETURN/ENTER to continue.

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

Experience shows that obtaining an asset via purchasing is always less in the long term than leasing, and since the various parameters in which leasing offers an advantage are not applicable in your case, the financial expenditure recommendation is to purchase the asset.
** End - RETURN/ENTER to continue

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

Would you care to analyze the financing for purchasing this particular asset?

> YES
  NO

1. Use the arrow key or first letter of item to position the cursor.
2. Press RETURN/ENTER to continue.

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

What is the exact or approximate cost of the asset?

25000

1. Enter a positive number.
2. Press RETURN/ENTER to continue.

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

What is the amount of the down payment?

1000

1. Enter a positive number.
2. Press RETURN/ENTER to continue.

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

How long in years is the length of the desired financing period?

4

1. Enter a positive number.
2. Press RETURN/ENTER to continue

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

What is the annual percentage rate for the financing?

10.5

1. Enter a positive number.
2. Press RETURN/ENTER to continue

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

```
The formulated annual asset payment is as follows: $6630

  ** End - RETURN/ENTER to continue
```

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

```
Would you care to analyze the financing of this asset
under other conditions?

  YES
> NO
```

OFFICE ASSET FINANCIAL EXPENDITURE DECISION SUPPORT SYSTEM

Conclusion:

Recommendation is as follows: PURCHASE the asset (90%)

Payment for the asset is as follows: $6630 PER YEAR

** End - RETURN/ENTER to continue

CHAPTER 6
CONCLUSION

6.1 Summary

We, have proposed a novel design approach for knowledge-based office task decision support based on a new office task goal decomposition model which incorporates knowledge engineering, artificial intelligence and pattern recognition techniques. The separation of control knowledge and domain knowledge from the hierarchical decomposition of goals we employ presents a more adequate model of the office domain than the classical procedural models for the following reasons: 1) Our approach is conceptually and computationally clean, in that it makes the representation of the office tasks modular and easily modifiable and extensible, since we employ goal frame structures for each office task activity to be supported.; 2) Formal (i.e. the structure is understood by the system) and informal (i.e. the structure is only understood by humans) descriptions of office work or task can be mixed. This means that users are not bothered to explain every action or step to the system.; 3) The possibility of local, domain-specific control-information at each level in the planning hierarchy introduces great potential in combining and managing subgoals. 4) The hierarchical goal frame-based

planning networks we employ provide a descriptive framework for indicating how tasks are to be organized. This framework represents goal-subgoal relationships that need to be known during execution monitoring and describes plans at various levels of detail.

We also propose a novel strategy for logically pruning an office task AND/OR tree object-rule base. The process we employ to conclude goal frames, is a somewhat radical departure from classical and contemporary models used in the office domain. Our novel approach to office task decision support offers promising results as a knowledge-based goal-driven strategy for computer-based support for office activities through microcomputer-based machine consultation and communication.

The structural aspects of the knowledge base incorporates three basic structures: goal frames, task parameters and object-rules. Conceptually, the over all operational strategy can be defined by three subprocesses: 1) generation; 2) utilization; and 3) revision; The approach we have proposed for knowledge-based system design for office task decision support can be applied to various tasks requiring intelligent decision support contingent on inclusive domain knowledge. We handle uncertainty in knowledge-based parameter through the introduction of what we call belief weighting factors.

6.2 Areas for Future Work

Suggestions for future research focus on two issues. 1) Defining strategies to support a subset of office task activities that are suitable for intelligent support system utilization for office automation via the definition and organization of office knowledge in terms of task parameters; and 2) the development of problem solving techniques germane to general office tasks execution for adaptive utilization. The first problem should be approached as pattern recognition feature extraction problem to determine task pattern parameters. The second problem should be viewed as a control problem that should be approached from an artificial intelligence frame of reference.

APPENDIX

SAMPLE AUTOSEC KNOWLEDGE-BASE EXAMPLE APPLICATIONS ILLUSTRATING GOAL FRAMES, TASK PARAMETERS, AND OBJECT-RULE BASES FOR THREE LISP IMPLEMENTATIONS: 1) PERSONNEL ASSIGNMENT DECISIONS; 2) MEETING SCHEDULING; AND 3) FINANCIAL EXPENDITURE PLANNING.

TASK Parameter Group GOAL FRAMETYPES

```
PERSONNEL [GOAL FRAMETYPES]
---------
TRANSLATION:  (PERSONNEL DECISION-MAKING FOR JOB APPLICANTS)
GOALS:  (POSITION)
DISPLAYRESULTS:  YES
PARMGROUP:  PERSONNEL-PARMS
RULEGROUPS:  (PERSONNEL-RULES)
IDENTIFIER:  "PERSONNEL- "
```

TASK Parameter Group PERSONNEL-PARMS

```
DEGREE [PERSONNEL-PARMS]
------
TRANSLATION:  (does job applicant possess a degree)
PROMPT:  (Does the job applicant possess a degree?)
TYPE:  SINGLEVALUED
EXPECT:  (YES NO)
USED-BY:  OR002 OR001 OR008 OR003

DISCOVERY [PERSONNEL-PARMS]
---------
TRANSLATION:  (checks to see if applicant made any important
discoveries)
PROMPT:  (Has the job applicant made any important
discoveries?)
TYPE:  SINGLEVALUED
EXPECT:  (YES NO)
USED-BY:  OR008 OR003
```

```
EXPERIENCE [PERSONNEL-PARMS]
----------
TRANSLATION:  (amount of experience the job applicant
possesses)
PROMPT:  (What amount experience in terms of years does the
          job applicant possess?)
EXPECT:  NUMBER
RANGE:  (1 20)
TYPE:  SINGLEVALUED
USED-BY:  OR004 OR005


GRADE [PERSONNEL-PARMS]
-----
TRANSLATION:  (grade point average of job applicant)
PROMPT:  (What is the grade point average of the job
          applicant?)
EXPECT:  NUMBER
RANGE:  (1 4)
TYPE:  SINGLEVALUED
USED-BY:  OR004 OR005 OR008 OR006


POSITION [PERSONNEL-PARMS]
--------
TRANSLATION:  (the type of position, if any, the job
               applicant is qualified for)
LEGALVALUES:  TEXT
TYPE:  SINGLEVALUED
UPDATED-BY:  OR004 OR005 OR001 OR008 OR003 OR006


QUALIFY [PERSONNEL-PARMS]
-------
TRANSLATION:  (denotes that the job applicant qualifies for
a position)
TYPE:  SINGLEVALUED
EXPECT:  (YES NO)
USED-BY:  OR004 OR005 OR006
UPDATED-BY:  OR002


TYPE-DEGREE [PERSONNEL-PARMS]
-----------
TRANSLATION:  (the type of degree the applicant possesses)
PROMPT:  (What type of degree of degree does the applicant
possess?)
EXPECT:  SINGLE-LINE-INPUT
TYPE:  ASK-ALL
```

AUTOSEC System Variables

```
$$TITLE [VARIABLES]
-------
VALUE:  MPRINTT :LEFT 3 :RIGHT 80 :LINE 1 :TAB 27 :ATTR QUOTE
(CYAN) AUTOSEC::LINE 2 :TAB 5 A KNOWLEDGE-BASED GOAL-DRIVEN
DECISION SUPPORT SYSTEM :LINE 1 :TAB 22 FOR OFFICE AUTOMATION

DOMAIN [VARIABLES]
------
VALUE:  "       PERSONNEL ASSIGNMENT          "
```

AUTOSEC System parameters

```
PERSONNEL-RULES [RULEGROUPS]
---------------
FRAME:  (PERSONNEL)
SVAL:  (THE PERSONNEL)
VALUE:  OR001 OR002 OR003 OR004 OR005 OR006 OR008
```

AUTOSEC Task parameters

```
FRAMETYPES [PARMGROUPS]
----------
VALUE:  PERSONNEL

PERSONNEL-PARMS [PARMGROUPS]
---------------
VALUE:  DEGREE DISCOVERY EXPERIENCE GRADE POSITION QUALIFY
TYPE-DEGREE
Object-Rule Group PERSONNEL-RULES


OR001 [PERSONNEL-RULES]
-------

PATTERN:  ($AND
            (SAME GOAL FRAME DEGREE NO))

ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME POSITION "THERE CURRENTLY IS
            NO MATCH BETWEEN AN AVALIABLE POSITION AND THE
            JOB APPLICANTS QUALIFICATIONS" TALLY 100))


OR002 [PERSONNEL-RULES]
-------
```

```
PATTTERN:  ($AND
             (SAME GOAL FRAME DEGREE YES))

ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME QUALIFY YES TALLY 100))

OR003 [PERSONNEL-RULES]
-------

PATTERN:  ($AND
            (SAME GOAL FRAME DEGREE YES)
            (SAME GAOL FRAME DISCOVERY YES))

ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME POSITION "JOB APPLICANT
             QUALIFIES FOR A RESEARCH POSITION" TALLY 100))

OR004 [PERSONNEL-RULES]
-------

PATTERN:  ($AND
            (SAME GOAL FRAME QUALIFY YES)
            (LESSP*
              (VAL1 GOAL FRAME GRADE ) 3.5)
            (GREATEQ*
              (VAL1 GOAL FRAME EXPERIENCE ) 2))

ACTION:  (DO-ALL
           (CONCLUDE FRAME POSITION "JOB APPLICANT IS
             RECOMMENDED FOR A POSITION IN SERVICE
              ENGINEERING" TALLY 100))


OR005 [PERSONNEL-RULES]
-------

PATTERN:  ($AND
            (SAME GOAL FRAME QUALIFY YES)
            (LESSP*
              (VAL1 GOAL FRAME GRADE ) 3.5)
            (LESSP*
              (VAL1 GOAL FRAME EXPERIENCE ) 2))

ACTION:  (DO-ALL
           (CONCLUDE FRAME POSITION "JOB APPLICANTS
             EXPERIECNCE AND ACADEMIC BACKGROUND DON NOT
              MATCH ANY AVAILABLE POSITIONS" TALLY 100))


OR006 [PERSONNEL-RULES]
-------
```

```
PATTERN:  ($AND
            (SAME GOAL FRAME QUALIFY YES)
            (GREATEQ*
              (VAL1 GOAL FRAME GRADE ) 3.5)
            (LESSEQ*
              (VAL1 GOAL FRAME GRADE) 3.7))

ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME POSITION "JOB APPLICANT
            QUALIFIES FOR A POSITION IN PRODUCT ENGINEERING"
             TALLY 100))

OR007 [PERSONNEL-RULES]
-------

PATTERN:  ($AND
            (SAME FRAME DEGREE YES)
            (GREATERP*
              (VAL1 FRAME GRADE) 3.7)
            (SAME FRAME DISCOVERY NO))
ACTION:  (DO-ALL
           (CONCLUDE FRAME POSITION "DATA INSUFFICIENT TO
            MAKE RECOMMENDATION"
            TALLY 100))


AUTOSEC System parameters

PERSONNEL-RULES [RULEGROUPS]
----------------
GOAL FRAME:  (PERSONNEL)
SVAL:  (THE PERSONNEL)
VALUE:  OR001 OR002 OR003 OR004 OR005 OR006 OR007


GOAL FRAMETYPES [PARMGROUPS]
-----------
VALUE:  PERSONNEL

PERSONNEL-PARMS [PARMGROUPS]
---------------
VALUE:  DEGREE DISCOVERY EXPERIENCE GRADE POSITION QUALIFY
        TYPE-DEGREE


Task Parameter Group FRAMETYPES

SCHEDULE [GOAL FRAMETYPES]
--------
```

```
TRANSLATION:  (root frame for scheduling domain)
IDENTIFIER:  "SCHEDULE-"
RULEGROUPS:  (SCHEDULE-RULES)
PARMGROUP:  SCHEDULE-PARMS
DISPLAYRESULTS:  YES
GOALS:  (SCHEDULE-MEETING)


TASK Parameter Group SCHEDULE-PARMS


DAY [SCHEDULE-PARMS]
---
TRANSLATION:  (day of meeting)
PROMPT:  (What DAY of the month is desired for the meeting?)
BELIEF WEIGHTING FACTOR:  POSITIVE
CONTAINED-IN:  (OR012 OR004)
TYPE:  ASK-ALL
EXPECT:  (1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
          21 22 23 24 25 2627 28 29 30 31)
USED-BY:  OR012 OR004
UPDATED-BY:  SREFMARK

HOUR [SCHEDULE-PARMS]
----
TRANSLATION:  (hour of meeting)
PROMPT:  (Specify desired HOUR of day for meeting to take
place:)
BELIEF WEIGHTING FACTOR:  POSITIVE
CONTAINED-IN:  (OR004 OR009)
TYPE:  ASK-ALL
EXPECT:  (1 2 3 4 5 6 7 8 9 10 11 12)
USED-BY:  OR004 OR009
UPDATED-BY:  SREFMARK

LOCATION [SCHEDULE-PARMS]
--------
TRANSLATION:  (location of meeting)
PROMPT:  (What is the desired meeting LOCATION?)
CONTAINED-IN:  (OR004)
EXPECT:  (CIR-CONF-ROOM LAB PROF-OFFICE)
TYPE:  ASK-ALL
UPDATED-BY:  SREFMARK


MINUTES [SCHEDULE-PARMS]
-------
TRANSLATION:  (minutes time component for meeting)
PROMPT:  (Specify MINUTES component of hour for time of
meeting:)
CONTAINED-IN:  (OR004 OR009)
TYPE:  ASK-ALL
```

```
EXPECT:  (10 15 20 25 30 35 40 45 50 55 60)
USED-BY:  OR004 OR009
UPDATED-BY:  SREFMARK


MONTH [SCHEDULE-PARMS]
-----
TRANSLATION:  (month of meeting)
PROMPT:  (What is the desired MONTH for the meeting?)
CERTAINTY-FACTOR-RANGE:  POSITIVE
CONTAINED-IN:  (OR008 OR004)
TYPE:  ASK-ALL
EXPECT:  (JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC)
USED-BY:  OR008 OR004
UPDATED-BY:  SREFMARK

PARTICIPANTS [SCHEDULE-PARMS]
------------
TRANSLATION:  (meeting participants)
PROMPT:  (Please enter "name(s)" of one or more persons who
         are meeting participants:)
CONTAINED-IN:  (OR006 OR004)
TYPE:  SINGLEVALUED
EXPECT:  SINGLE-LINE-INPUT
UPDATED-BY:  SREFMARK

PURPOSE [SCHEDULE-PARMS]
-------
TRANSLATION:  (purpose of meeting)
PROMPT:  (Please enter purpose of meeting:)
CONTAINED-IN:  (OR004)
TYPE:  SINGLEVALUED
EXPECT:  SINGLE-LINE-INPUT
LEGALVALUES:  ANY
UPDATED-BY:  SREFMARK

SCHEDULE-MEETING [SCHEDULE-PARMS]
----------------
TRANSLATION:  (schedule)
LEGALVALUES:  ANY
TYPE:  MULTIVALUED
USED-BY:  OR006
UPDATED-BY:  OR008 OR012 OR004 OR009 SREFMARK OR006

TIME-TYPE [SCHEDULE-PARMS]
---------
TRANSLATION:  (am or pm time)
PROMPT:  (Is this time am or pm?)
CONTAINED-IN:  (RULE004 RULE009)
TYPE:  ASK-ALL
EXPECT:  (AM PM)
USED-BY:  OR004 OR009
```

```
UPDATED-BY:  SREFMARK


TXT3 [SCHEDULE-PARMS]
----
TRANSLATION:  (at)
TYPE:  SINGLEVALUED
UPDATED-BY:  SREFMARK

TXT4 [SCHEDULE-PARMS]
----
TRANSLATION:  (:)
TYPE:  SINGLEVALUED
UPDATED-BY:  SREFMARK

TXT5 [SCHEDULE-PARMS]
----
TRANSLATION:  (in the)
TYPE:  SINGLEVALUED
UPDATED-BY:  SREFMARK

TXT6 [SCHEDULE-PARMS]
----
TRANSLATION:  (the purpose of which is)
TYPE:  SINGLEVALUED
UPDATED-BY:  SREFMARK

TXT7 [SCHEDULE-PARMS]
----
TRANSLATION:  (the meeting participants are)
TYPE:  SINGLEVALUED


TASK Parameter Group TEXTAGS


TXT10 [TEXTAGS]
-----
TRANSLATION:  (as the desired time for the meeting, please
              CONTINUE and select a new TIME at the
              appropriate prompt.)
TYPE:  TEXTAG

TXT11 [TEXTAGS]
-----
TRANSLATION:  (as the desired DAY of the month for the
               meeting, please CONTINUE and select a new day
               for month for the meeting to take place.)
TYPE:  TEXTAG

TXT9 [TEXTAGS]
----
```

```
TRANSLATION:  (as the desired month for the meeting, please
               CONTINUE and select another MONTH for the
                meeting at the appropriate prompt.)
TYPE:  TEXTAG


AUTOSEC SYSTEM Variables


$$MESSAGE [VARIABLES]
---------
INTEGER:  (1. You must enter a positive integer 2. Press
              RETURN/ENTER to continue)
VALUE:  #!TRUE

$$TITLE [VARIABLES]
-------
VALUE:  MPRINTT :LEFT 3 :RIGHT 80 :LINE 1 :TAB 27 :ATTR QUOTE
        (CYAN) AUTOSEC::LINE 2 :TAB 5 A KNOWLEDGE-BASED
        GOAL-DRIVEN DECISION SUPPORT SYSTEM
        :LINE 1 :TAB 22 FOR OFFICE AUTOMATION

DOMAIN [VARIABLES]
------
VALUE:  "          Meeting Scheduling       "

OWNER-DAY [VARIABLES]
---------
VALUE:  ?

OWNER-HOUR [VARIABLES]
----------
VALUE:  ?

OWNER-MINUTES [VARIABLES]
-------------
VALUE:  ?

OWNER-MONTH [VARIABLES]
-----------
VALUE:  ?

OWNER-TIME-TYPE [VARIABLES]
---------------
VALUE:  ?


AUTOSEC System parameters


SCHEDULE-RULES [RULEGROUPS]
--------------
```

```
SVAL:  (THE SCHEDULE)
GOAL FRAME:  (SCHEDULE)
VALUE:  OR004 OR006 OR008 OR009 OR012


    System parameters


FRAMETYPES [PARMGROUPS]
----------
VALUE:  SCHEDULE

SCHEDULE-PARMS [PARMGROUPS]
--------------
VALUE:  DAY HOUR LOCATION MINUTES MONTH PARTICIPANTS PURPOSE
        SCHEDULE-MEETING TIME-TYPE TXT3 TXT4 TXT5 TXT6 TXT7

Object-Rule Group SCHEDULE-RULES


OR001 [SCHEDULE-RULES]
-------

PATTERN:  ($AND
            (SAME GOAL FRAME MONTH
              (VALUE-OF OWNER-MONTH))
            (SAME GOAL FRAME DAY
              (VALUE-OF OWNER-DAY))
            (SAME GOAL FRAME HOUR
              (VALUE-OF OWNER-HOUR))
            (SAME GOAL FRAME MINUTES
              (VALUE-OF OWNER-MINUTES))
            (SAME GOAL FRAME TIME-TYPE
              (VALUE-OF OWNER-TIME-TYPE)))
ACTION:  (DO-ALL
           (CONCLUDE FRAME SCHEDULE-MEETING
             (TEXT  "The meeting will take place on"
               (VAL1 GOAL FRAME MONTH )
               (VAL1 GOAL FRAME DAY )
               (TEXT TXT3)
               (VAL1 GOAL FRAME HOUR )
               (TEXT TXT4)
               (VAL1 GOAL FRAME MINUTES )
               (VAL1 GOAL FRAME TIME-TYPE )
               (TEXT TXT5)
               (VAL1 GOAL FRAME LOCATION )
               (TEXT TXT6)
               (VAL1 GOAL FRAME PURPOSE )
               (TEXT TXT7)
               (VAL1 GOAL FRAME PARTICIPANTS )) TALLY 100))
```

```
OR002 [SCHEDULE-RULES]
-------

PATTERN:  ($AND
            (NOTKNOWN GOAL FRAME SCHEDULE-MEETING))
ACTION:   (DO-ALL
            (CONCLUDE GOAL FRAME SCHEDULE-MEETING
              (TEXT  "there currently exists insufficient
                      information to make a decision for
                      scheduling a meeting for"
                (VAL1 GOAL FRAME PARTICIPANTS )) TALLY 100))


OR003 [SCHEDULE-RULES]
-------

PATTERN:  ($AND
            (NOTSAME GOAL FRAME MONTH
              (VALUE-OF OWNER-MONTH)))
ACTION:   (DO-ALL
            (CONCLUDE GOAL FRAME SCHEDULE-MEETING
              (TEXT  "there exists a conflict with"
                (VAL1 GOAL FRAME MONTH )
                (TEXT TXT9)) TALLY 100))

OR004 [SCHEDULE-RULES]
-------

PATTERN:  ($AND
            ($OR
              (NOTSAME GOAL FRAME HOUR
                (VALUE-OF OWNER-HOUR))
              (NOTSAME GOAL FRAME MINUTES
                (VALUE-OF OWNER-MINUTES))
              (NOTSAME GOAL FRAME TIME-TYPE
                (VALUE-OF OWNER-TIME-TYPE))))
ACTION:   (DO-ALL
            (CONCLUDE GOAL FRAME SCHEDULE-MEETING
              (TEXT  "there exists a conflict with"
                (VAL1 GOAL FRAME HOUR )
                (TEXT TXT4)
                (VAL1 GOAL FRAME MINUTES )
                (VAL1 GOAL FRAME TIME-TYPE )
                (TEXT TXT10)) TALLY 100))

OR005 [SCHEDULE-RULES]
-------

PREMISE:  ($AND
            (NOTSAME GOAL FRAME DAY
              (VALUE-OF OWNER-DAY)))
```

```
ACTION:  (DO-ALL
            (CONCLUDE GOAL FRAME SCHEDULE-MEETING
              (TEXT  "there exist a conflict with"
                (VAL1 GOAL FRAME DAY )
                (TEXT TXT11)) TALLY 100))


 AUTOSEC  System parameters

 SCHEDULE-RULES [RULEGROUPS]
 ---------------
 SVAL:  (THE SCHEDULE)
 GOAL FRAME:  (SCHEDULE)
 VALUE:  OR004 OR006 OR008 OR009 OR012


 GOAL FRAMETYPES [PARMGROUPS]
 ----------
 VALUE:  SCHEDULE


 SCHEDULE-PARMS [PARMGROUPS]
 ---------------
 VALUE:  DAY HOUR LOCATION MINUTES MONTH PARTICIPANTS PURPOSE
 SCHEDULE-MEETING TIME-TYPE TXT3 TXT4 TXT5 TXT6 TXT7



 TASK Parameter Group ASSET-PARMS


 ACQUIRE-BY [ASSET-PARMS]
 ----------
 TRANSLATION:  (determination how to acquire the asset)
 TYPE:  SINGLEVALUED
 USED-BY:  OR005 OR006
 UPDATED-BY:  OR001 OR002 SREFMARK OR004

 CANNOT-BORROW [ASSET-PARMS]
 -------------
 TRANSLATION:  (your credit is too low for you to get a loan)
 TYPE:  YES/NO
 USED-BY:  OR002
 UPDATED-BY:  OR001

 CASH-RESERVE-NEEDED [ASSET-PARMS]
 -------------------
 TRANSLATION:  (do you need to maintain large cash reserves)
 PROMPT:  (In your business, do you need to maintain larger
           than average cash reserves to take advantage of
            unexpected opportunities?)
 ASKFIRST:  YES
```

```
TYPE:  YES/NO
USED-BY:  OR003

HOW-TO-ACQUIRE [ASSET-PARMS]
---------------
TRANSLATION:  (my recommendation)
TYPE:  MULTIVALUED
LEGALVALUES:  TEXT
USED-BY:  OR004
UPDATED-BY:  OR002 SREFMARK OR004

LESSE-CASH [ASSET-PARMS]
-----------
TRANSLATION:  (your cash reserves)
PROMPT:  (How would you describe your company's cash
reserves?)
ASKFIRST:  YES
TYPE:  SINGLEVALUED
EXPECT:  (GOOD FAIR POOR)
USED-BY:  OR003

LESSE-CREDIT [ASSET-PARMS]
------------
TRANSLATION:  (your credit rating)
PROMPT:  (How would you describe your company's current
credit rating?)
HELP:  (GOOD - you could borrow enough to buy the asset, and
the extra debt would not significantly lower your rating;
FAIR - you could borrow, and the extra debt would
significantly lower your credit rating; POOR
- you could not borrow enough to buy the asset.)
BELIEF-WEIGHTING-FACTOR:  POSITIVE
TYPE:  SINGLEVALUED
EXPECT:  (GOOD FAIR POOR)
USED-BY:  OR003 OR001

PAYMENT [ASSET-PARMS]
-------
TRANSLATION:  (payment for the asset)
TYPE:  SINGLEVALUED
UPDATED-BY:  OR007

PRESERVES-CASH [ASSET-PARMS]
--------------
TRANSLATION:  (a lease would preserve your cash reserves)
TYPE:  YES/NO
USED-BY:  OR002
UPDATED-BY:  OR003


TASK  Parameter Group FINANCE-PARMS
```

```
ASSET-COST [FINANCE-PARMS]
----------
PROMPT:  (What is the cost of the asset?)
EXPECT:  POSITIVE-NUMBER
TYPE:  SINGLEVALUED
CONTAINED-IN:  (OR005 OR006)

DOWN-PAYMENT [FINANCE-PARMS]
------------
PROMPT:  (What is the amount of the down payment?)
EXPECT:  POSITIVE-NUMBER
TYPE:  SINGLEVALUED
CONTAINED-IN:  (OR005)

FINANCE-INTEREST [FINANCE-PARMS]
----------------
PROMPT:  (What is the percent yield to the firm for the
financing?)
EXPECT:  NUMBER
TYPE:  SINGLEVALUED
CONTAINED-IN:  (OR005 OR006)

FINANCE-IT [FINANCE-PARMS]
----------
TRANSLATION:  (annual payment)
TYPE:  SINGLEVALUED
CONTAINED-IN:  (OR007)
USED-BY:  OR007
UPDATED-BY:  OR005 OR006

FINANCE-PERIOD [FINANCE-PARMS]
--------------
PROMPT:  (What is the length in years of the financing
period?)
EXPECT:  POSITIVE-NUMBER
TYPE:  SINGLEVALUED
CONTAINED-IN:  (OR005 OR006)


TASK Parameter Group GOAL FRAMETYPES


ASSET [FRAMETYPES]
-----
TRANSLATION:  (a demostration system which reflects a part of
a lease versus buy decision support system)
IDENTIFIER:  "ASSET-"
RULEGROUPS:  (ASSET-RULES)
PARMGROUP:  ASSET-PARMS
DISPLAYRESULTS:  YES
GOALS:  (HOW-TO-ACQUIRE PAYMENT)
INITIALDATA:  (LESSE-CREDIT)
```

```
OFFSPRING:  (FINANCE)

FINANCE [FRAMETYPES]
-------
TRANSLATION:  (analyzing the financing)
PROMPT2ND:  (Would you care to analyze the financing for the
asset under other conditions?)
PROMPT1ST:  (Would you care to analyze the financing for this
asset?)
GOALS:  (FINANCE-IT)
PARENTS:  (ASSET)
PARMGROUP:  FINANCE-PARMS
RULEGROUPS:  (FINANCE-RULES)
IDENTIFIER:  "FINANCE-"
DISPLAYRESULTS:  YES

TASK Parameter Group TEXTAGS


TXTG1 [TEXTAGS]
-----
TRANSLATION:  (LEASE the asset)
TYPE:  TEXTAG

TXTG2 [TEXTAGS]
-----
TRANSLATION:  (BUY the asset)
TYPE:  TEXTAG

AUTOSEC System Variables


$$TITLE [VARIABLES]
-------
VALUE:  MPRINTT :LEFT 3 :RIGHT 80 :LINE 1 :TAB 27 :ATTR QUOTE
(CYAN) AUTOSEC::LINE 2 :TAB 5 A KNOWLEDGE-BASED GOAL-DRIVEN
DECISION SUPPORT SYSTEM :LINE 1 :TAB 22 FOR OFFICE AUTOMATION

DOMAIN [VARIABLES]
------
VALUE:  "        OFFICE ASSET LEASE OR BUY DECISION SUPPORT
SYSTEM      "


AUTOSEC System parameters


ASSET-RULES [RULEGROUPS]
-----------
SVAL:  (THE ASSET)
FRAME:  (ASSET)
VALUE:  OR001 OR002 OR003 OR004
```

```
FINANCE-RULES [RULEGROUPS]
--------------
FRAME:  (FINANCE)
SVAL:  (THE FINANCE)
VALUE:  OR005 OR006 OR007


AUTOSEC Task parameters


ASSET-PARMS [PARMGROUPS]
-----------
VALUE:  ACQUIRE-BY CANNOT-BORROW CASH-RESERVE-NEEDED
HOW-TO-ACQUIRE LESSE-CASH LESSE-CREDIT PAYMENT PRESERVES-CASH

FINANCE-PARMS [PARMGROUPS]
-------------
VALUE:  ASSET-COST DOWN-PAYMENT FINANCE-INTEREST FINANCE-IT
FINANCE-PERIOD

GOAL FRAMETYPES [PARMGROUPS]
----------
VALUE:  FINANCE ASSET
Object-Rule Group ASSET-RULES


OR001 [ASSET-RULES]
-------

PATTERN:  ($AND
            (SAME GOAL FRAME LESSE-CREDIT POOR))
ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME CANNOT-BORROW YES TALLY 100)
           (CONCLUDE GOAL FRAME ACQUIRE-BY LEASE TALLY 100)
           (MPRINTT "Your credit is not adequate; you cannot
                     borrow money to buy the asset,
                     therefore, LEASE the asset."))

0R002 [ASSET-RULES]
-------

PATTERN:  ($AND
            ($OR
              (SAME GOAL FRAME CANNOT-BORROW)
              (SAME GOAL FRAME PRESERVES-CASH)))
ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME HOW-TO-ACQUIRE
             (TEXT TXTG1) TALLY 100)
           (CONCLUDE FRAME ACQUIRE-BY LEASE TALLY 100))

OR003 [ASSET-RULES]
-------
```

```
PATTERN:  ($AND
            (SAME GOAL FRAME LESSE-CREDIT FAIR)
            (SAME GOAL FRAME LESSE-CASH FAIR)
            (SAME GOAL FRAME CASH-RESERVE-NEEDED))
ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME PRESERVES-CASH YES TALLY 100)
           (MPRINTT "Your credit rating will be affected
                     regardless of whether you buy or lease
                     this asset.  However, your business
                     needs larger than average cash reserves,
                     which a down payment on a loan would
                     deplete.  Therefore, a lower front-end
                     cost of a LEASE will maintain your cash
                     reserves."))


OR004 [ASSET-RULES]
-------

PATTERN:  ($AND
            (NOTKNOWN GOAL FRAME HOW-TO-ACQUIRE))
ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME HOW-TO-ACQUIRE
             (TEXT TXTG2) TALLY 100)
           (CONCLUDE GOAL FRAME ACQUIRE-BY PURCHASE TALLY
                100)
           (MPRINTT "Since experience shows that buying is
             almost always cheaper than leasing, and since
             the special cases in which leasing offers an
              advantage do not seem to apply in your case,
              BUY the asset."))


Object-Rule Group FINANCE-RULES


OR005 [FINANCE-RULES]
-------

PATTERN:  ($AND
            (SAME GOAL FRAME ACQUIRE-BY PURCHASE))
ACTION:  (DO-ALL
           (CONCLUDE GOAL FRAME FINANCE-IT
             (TIMES
               (FQUOTIENT
                 (DIFFERENCE
                   (VAL1 GOAL FRAME ASSET-COST )
                   (VAL1 GOAL FRAME DOWN-PAYMENT ))
                 (VAL1 GOAL FRAME FINANCE-PERIOD ))
               (FQUOTIENT
                 (PLUS 100
```

```
                    (VAL1 GOAL FRAME FINANCE-INTEREST )) 100))
TALLY 100))

OR006 [FINANCE-RULES]
-------

PATTERN:  ($AND
            (SAME GOAL FRAME ACQUIRE-BY LEASE))
ACTION:   (DO-ALL
            (CONCLUDE GOAL FRAME FINANCE-IT
              (PLUS
                (PAY-CALC ASSET-COST FINANCE-INTEREST
                          FINANCE-PERIOD)
                (FQUOTIENT
                  (TIMES
                    (VAL1 GOAL FRAME ASSET-COST ) 2) 100))
                   TALLY 100))

OR007 [FINANCE-RULES]
-------

PATTERN:  ($AND
            (KNOWN GOAL FRAME FINANCE-IT))
ACTION:   (DO-ALL
            (CONCLUDE GOAL FRAME PAYMENT
              (TEXT  "$"
                (VAL1 GOAL FRAME FINANCE-IT ) "PER YEAR")
                  TALLY 100))
```

User defined Functions

```
PAY-CALC [FUNCTIONS]
--------
TEMPLATE:  (PARM PARM PARM)
TYPE:  EXPRESSION
TRANSLATION:  (% 1 *
                 (% 2) / 100 +
                 (% 1) /
                 (% 3))
SOURCE:   (LAMBDA
            (ASSET-COST FINANCE-INTEREST FINANCE-PERIOD)
            (PLUS
              (FQUOTIENT
                (TIMES
                  (VAL1 FRAME ASSET-COST)
                  (VAL1 FRAME FINANCE-INTEREST)) 100)
              (FQUOTIENT
                (VAL1 FRAME ASSET-COST)
                (VAL1 FRAME FINANCE-PERIOD))))
```

AUTOSEC System parameters

```
ASSET-RULES [RULEGROUPS]
-----------
SVAL:  (THE ASSET)
GOAL FRAME:  (ASSET)
VALUE:  OR001 OR002 OR003 OR004

FINANCE-RULES [RULEGROUPS]
-------------
GOAL FRAME:  (FINANCE)
SVAL:  (THE FINANCE)
VALUE:  OR005 OR006 OR007

ASSET-PARMS [PARMGROUPS]
-----------
VALUE:  ACQUIRE-BY CANNOT-BORROW CASH-RESERVE-NEEDED
     HOW-TO-ACQUIRE LESSE-CASH LESSE-CREDIT PAYMENT
     PRESERVES-CASH

FINANCE-PARMS [PARMGROUPS]
-------------
VALUE:  ASSET-COST DOWN-PAYMENT FINANCE-INTEREST FINANCE-IT
        FINANCE-PERIOD

GOAL FRAMETYPES [PARMGROUPS]
----------
VALUE:  FINANCE ASSET
```

REFERENCES

Barber, G. R., "Supporting Organizational Problem-solving with a Workstation," ACM Trans. on Office Information Systems, Vol. 1, 1983.

Barr, A., and Feigenbaum, E., eds., The Handbook of AI, Vol. I, W. Kaufman, Inc., Los Altos, California., 1981.

Barr, A., and Feigenbaum, E., eds., The Handbook of AI, Vol. II, W. Kaufman, Inc., Los Altos, California., 1982.

Bracchi, G., and Pernici, B., "S.O.S.: A Conceptual Model for Office Informatiom Systems," Proc. of ACM AIGMOD on Databases, 1983.

Buchanan, B. G. and Fiegenbaum, E. A., "Dendral and Meta-Dendral: Their Applications Dimension, Artificial Intelligence. Vol. 11, 1978.

Buchanan, B. G. and Shortliffe, E. H., Rule-based Expert Systems. Addision-Wesley Publishing Company, Reading, Massachusetts, 1984.

Chang, L. C., and Tou, J. T., "MEDIKS- A Medical Knowledge System," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-14, No. 5, Sept./Oct., pp. 746-750, 1984.

Croft, W. B., Lefkowitz, L. S., "Task Support in an Office System," ACM Trans. on Office Information Systems, Vol. 2,No. 3, 1984.

Croft, W. B., and Leftkowitz, L. S., "Using a Planner to Support Office Work," ACM-IEEE Joint Conference on Office Information Systems, 1988.

Dai, M. T., and Tou, J. T,. "KETOG - A Knowledge Engineering Tool for Organizing and Constructing the Knowledge Base in Pattern-Directed Expert Systems," Proceedings of the International Computer Symposium, pp. 647-652, 1988.

Ellis, C. A., "Information control nets: a mathematical model of office information flow," ACM Proc. of Conf. on Simulation, Modeling and Measurement of Computer Systems, 1979.

Ellis, C. A., and Bernal, M., "OFFICETALK-D: An Experimental Office Information System," ACM SIGOA Conference on Office Systems, 1982.

Jong, D., "The System for Business Automation (SBA): A Unified Application Development System," Information Processing 80, North-Holland Publishing Co., Ed. Lavington, S. H., New York, 1980.

Kratzer, K., "The Desktop Metaphor: Visualization of Office Applications," IEEE-CS Symposium on OA, 1987.

Kunin, J. S., "OSL: An Office Specification Language," MIT, Boston, Massachusettes, 1981.

Lochovsky, F. H., "Managing Office Tasks," IEEE-CS Symposium on OA, 1987.

Maes, P., "Goals in Knowledge-based Office Systems," Proc. 8th German Workshop on AI, Wignst/Stade, 1984.

Maiochi, R., and Pernici, B., "Verification and Refinement of Office Procedures," IEEE-CS Symposium on OA, 1987.

Mazer, M. S., "Exploring the use of Distributed Problem Solving in Office Support Systems", IEEE OAS, 1987.

Minsky, M., "A Framework for Representing Knowledge," P. H. Winston (ed), The Psychology of Computer Vison, McGraw-Hill, New York, 1975.

Patrick, E. A., and Fattu, J. M., Artificial Intelligence with Statistical Pattern Recognition, Prentice-Hall, Inc., Englewood-Cliffs, New Jersey, 1986.

Rich, Elaine, Artificial Intelligence, McGraw-Hill, 1983.

Safayeri, F. R., Purdy, R. L., and Higgins, C. A., "Personal Computers in the Organizational Context," IEEE-CS Symposium on OA, 1987.

Shortliffe, E. H., Buchanan, B. G., and Feigenbaum, E. A., Knowledge Engineering for Medical Decision-making: A Review of Computer-based Clinical Decision Aids. Proceedings of the IEEE, 1977, Vol. 67, No. 9, 1207-1224.

Simon, H. A., "The Structure of ILL-Structured Problems," Artificial Intelligence, Vol. 4, 1973.

Slage, J. R., and Gayer, M. W., "An Intelligent Control Strategy for Computer Consultation," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. , 1984.

Suh, S., A Knowledge-based Consultation System for Agriculture. Master's Thesis, University of Florida, Gainesville, Florida, 1981.

Texas Instruments, Inc., Personal Consutant-Plus Technical Reference, AI Laboratory, Dallas,Texas 1986.

Tou, J. T., "MEDIKS- A Medical Knowledge System Design," Proc. on the 31st Annual Conference on Engineering in Medicine, 1978.

Tou, J. T., "Application of Pattern Recognition to Knowledge System Design and Diagnostic Inference," Pattern Recognition Theory and Applications, J. Kittler, K. S. Fu, and L. F. Pau (Eds), pp. 413-429, D. Riedel Publishing Co., New York, N. Y., 1982.

Tou, J. T., "Knowledge Engineering and Knowledge-based Systems," International Computer Symposium, Keynote Speech, Taiwan, ROC, 1984.

Tou, J. T., "Knowledge Engineering Revisited," International Journal of Computer and Informational Sciences, Vol. 14, No. 3, 1985.

Tou, J. T., and Cheng, J. M., "Design of a Computer-based Expert System for Applications in Agriculture," Proc. of the IEEE Symp. Automating Intelligent Behavior, 1983.

Tou, J. T., and Gonzalez, R. C., Pattern Recognition Principles, Addison-Wesley, 1974.

Tou, J. T., Huang, C. L., and Li, W. H., "Design of a Knowledged-based System for Understanding Electronic Circuit Diagrams," Proc. 1st Conf. on AI Applications, 1984.

Trischritzis, D., "A Form Manipulation System," Proceedings of the NYU Symposium on Automated Office Systems, NYU, 1979.

Trischritzis, D., and Gibbs, S., "Messages, Messengers, and objects," IEEE-CS Symposium on OA, 1987.

Tueni, M., Li, J, and Fares, P., "AMS: A Knowledge-based Approach to Task Representation, Organization and Coordination," ACM-IEEE Joint Conference on Office Information Systems, 1988.

Waterman, D. A., A Guide to Expert Systems, Addision-Wesley, Reading,Massachusetts, 1986.

Wier, Micheal, Goal-Directed Behaviour, Studies in Cybernetics:6, Gordon and Breach Science Publishers, 1984.

Wilson, R. D., Knowledge-based System Design Using APRIKS, Master's Thesis, EE dept., Center for Information Research, University of Florida, Gainesville, Florida, 1986.

Winston, P. H., Artificial Intelligence. 2nd ed. Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.

Zisman, M. D., "Representation, specification and automation of office procedures," Ph.D. Dissertation, University. of Pennsylvania, Philadelphia, Pennsylvania 1977.

Zloof, M. M., "Office by Example: A Business Language that Unifies Data and Word Processing and Electronic Mail," IBM Systems Journal, Vol. 1, No. 3, 1982.

BIOGRAPHICAL SKETCH

Roderick D. Wilson was born in Baton Rouge, Louisiana, on August 8, 1960. In August 1982, he received the degree of Bachelor of Science in electrical engineering from Southern University, after being awarded a four-year academic scholarship. Upon graduation he was employed as a design engineer at Texas Instruments, Inc. from 1982-1984.

After receiving a graduate fellowship to pursue a master's degree, he enrolled at the University of Florida in the fall of 1984 and received his Master of Science degree in electrical engineering, computer and information engineering option, in the spring of 1986. During his matriculation through the master's degree program he was employed as a both a hardware engineer II and software engineer II at Wang Laboratories, Inc.

Since the fall of 1986, Roderick has pursued work toward his doctoral degree in electrical engineering, computer and information engineering option in the Center for Information Research (CIR), after having been awarded a doctoral fellowship.

Roderick is a member of Tau Beta Pi, Eta Kappa Nu, IEEE, IEEE-CS and ACM. Upon graduation, he will be employed as a member of technical staff at Bell Communications Research. His

research interests are pattern recognition, artificial intelligence and knowledge-based systems, and digital communication networks.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Julius T. Tou, Chairman
Graduate Research Professor of
Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John Staudhammer, Co-chairman
Professor of Electrical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Keith L. Doty
Professor of Electrical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Zoran R. Pop-Stojanovic
Professor of Mathematics

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Fazil Najafi
Assitant Professor of Civil
Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

December 1990

Herbert A. Bruns

for Winfred M. Phillips
Dean, College of
Engineering

Madelyn M. Lockhart
Dean, Graduate School